Wireshark User's Guide

28442 for Wireshark 1.0.0

Ulf Lamping,
Richard Sharpe, NS Computer Software and Services P/L
Ed Warnicke,

Wireshark User's Guide: 28442

for Wireshark 1.0.0

by Ulf Lamping, Richard Sharpe, and Ed Warnicke

Copyright © 2004-2008 Ulf Lamping Richard Sharpe Ed Warnicke

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU General Public License, Version 2 or any later version published by the Free Software Foundation.

All logos and trademarks in this document are property of their respective owner.

Table of Contents

Preface ix
1. Foreword ix
2. Who should read this document? x
3. Acknowledgements xi
4. About this document xii
5. Where to get the latest copy of this document? xiii
6. Providing feedback about this document xiv
1. Introduction 1
1.1. What is Wireshark? 1
1.1.1. Some intended purposes 1
1.1.2. Features 1
1.1.3. Live capture from many different network media 2
1.1.4. Import files from many other capture programs 2
1.1.5. Export files for many other capture programs 2
1.1.6. Many protocol decoders 3
1.1.7. Open Source Software 3
1.1.8. What Wireshark is not 3
1.2. System Requirements 4
1.2.1. General Remarks 4
1.2.2. Microsoft Windows 4
1.2.3. Unix / Linux 5
1.3. Where to get Wireshark? 6
1.4. A brief history of Wireshark 7
1.5. Development and maintenance of Wireshark 8
1.6. Reporting problems and getting help 9
1.6.1. Website 9
1.6.2. Wiki 9
1.6.3. FAQ 9
1.6.4. Mailing Lists 9
1.6.5. Reporting Problems 10
1.6.6. Reporting Crashes on UNIX/Linux platforms 10
1.6.7. Reporting Crashes on Windows platforms 11
2. Building and Installing Wireshark 13
2.1. Introduction 13
2.2. Obtaining the source and binary distributions 14
2.3. Before you build Wireshark under UNIX 15
2.4. Building Wireshark from source under UNIX 18
2.5. Installing the binaries under UNIX 20
2.5.1. Installing from rpm's under Red Hat and alike 20
2.5.2. Installing from deb's under Debian 20
2.5.3. Installing from portage under Gentoo Linux 20
2.5.4. Installing from packages under FreeBSD 20
2.6. Troubleshooting during the install on Unix 21
2.7. Building from source under Windows 22
2.8. Installing Wireshark under Windows 23
2.8.1. Install Wireshark 23
2.8.2. Manual WinPcap Installation 25
2.8.3. Update Wireshark 25
2.8.4. Update WinPcap 25
2.8.5. Uninstall Wireshark 26
2.8.6. Uninstall WinPcap 26
3. User Interface 28
3.1. Introduction 28

3.2. Start Wireshark 29
3.3. The Main window 30
3.3.1. Main Window Navigation 31
3.4. The Menu 32
3.5. The "File" menu 34
3.6. The "Edit" menu 37
3.7. The "View" menu 39
3.8. The "Go" menu 44
3.9. The "Capture" menu 46
3.10. The "Analyze" menu 48
3.11. The "Statistics" menu 50
3.12. The "Tools" menu 53
3.13. The "Help" menu 54
3.14. The "Main" toolbar 56
3.15. The "Filter" toolbar 59
3.16. The "Packet List" pane 61
3.17. The "Packet Details" pane 62
3.18. The "Packet Bytes" pane 63
3.19. The Statusbar 64
4. Capturing Live Network Data 67
4.1. Introduction 67
4.2. Prerequisites 68
4.3. Start Capturing 69
4.4. The "Capture Interfaces" dialog box 70
4.5. The "Capture Options" dialog box 72
4.5.1. Capture frame 73
4.5.2. Capture File(s) frame 74
4.5.3. Stop Capture... frame 74
4.5.4. Display Options frame 75
4.5.5. Name Resolution frame 75
4.5.6. Buttons 75
4.6. The "Interface Details" dialog box 76
4.7. Capture files and file modes 77
4.8. Link-layer header type 79
4.9. Filtering while capturing 80
4.9.1. Automatic Remote Traffic Filtering 81
4.10. While a Capture is running 83
4.10.1. Stop the running capture 83
4.10.2. Restart a running capture 84
5. File Input / Output and Printing 86
5.1. Introduction 86
5.2. Open capture files 87
5.2.1. The "Open Capture File" dialog box 87
5.2.2. Input File Formats 89
5.3. Saving captured packets 92
5.3.1. The "Save Capture File As" dialog box 92
5.3.2. Output File Formats 94
5.4. Merging capture files 96
5.4.1. The "Merge with Capture File" dialog box 96
5.5. File Sets 98
5.5.1. The "List Files" dialog box 98
5.6. Exporting data 100
5.6.1. The "Export as Plain Text File" dialog box 100
5.6.2. The "Export as PostScript File" dialog box 101
5.6.3. The "Export as CSV (Comma Separated Values) File" dialog box 101
5.6.4. The "Export as C Arrays (packet bytes) file" dialog box 102
5.6.5. The "Export as PSML File" dialog box 102
5.6.6. The "Export as PDML File" dialog box 103

5.6.7. The "Export selected packet bytes" dialog box 104
5.6.8. The "Export Objects" dialog box 105
5.7. Printing packets 108
5.7.1. The "Print" dialog box 108
5.8. The Packet Range frame 110
5.9. The Packet Format frame 111
6. Working with captured packets 113
6.1. Viewing packets you have captured 113
6.2. Pop-up menus 115
6.2.1. Pop-up menu of the "Packet List" pane 115
6.2.2. Pop-up menu of the "Packet Details" pane 117
6.3. Filtering packets while viewing 120
6.4. Building display filter expressions 122
6.4.1. Display filter fields 122
6.4.2. Comparing values 122
6.4.3. Combining expressions 124
6.4.4. A common mistake 125
6.5. The "Filter Expression" dialog box 127
6.6. Defining and saving filters 129
6.7. Defining and saving filter macros 132
6.8. Finding packets 133
6.8.1. The "Find Packet" dialog box 133
6.8.2. The "Find Next" command 134
6.8.3. The "Find Previous" command 134
6.9. Go to a specific packet 135
6.9.1. The "Go Back" command 135
6.9.2. The "Go Forward" command 135
6.9.3. The "Go to Packet" dialog box 135
6.9.4. The "Go to Corresponding Packet" command 135
6.9.5. The "Go to First Packet" command 135
6.9.6. The "Go to Last Packet" command 135
6.10. Marking packets 136
6.11. Time display formats and time references 137
6.11.1. Packet time referencing 137
7. Advanced Topics 140
7.1. Introduction 140
7.2. Following TCP streams 141
7.2.1. The "Follow TCP Stream" dialog box 141
7.3. Expert Infos 143
7.3.1. Expert Info Entries 143
7.3.2. "Expert Info Composite" dialog 144
7.3.3. "Colorized" Protocol Details Tree 145
7.3.4. "Expert" Packet List Column (optional) 146
7.4. Time Stamps 147
7.4.1. Wireshark internals 147
7.4.2. Capture file formats 147
7.4.3. Accuracy 147
7.5. Time Zones 149
7.5.1. Set your computer's time correctly! 150
7.5.2. Wireshark and Time Zones 150
7.6. Packet Reassembling 152
7.6.1. What is it? 152
7.6.2. How Wireshark handles it 152
7.7. Name Resolution 154
7.7.1. Name Resolution drawbacks 154
7.7.2. Ethernet name resolution (MAC layer) 154
7.7.3. IP name resolution (network layer) 155
7.7.4. IPX name resolution (network layer) 155

7.7.5. TCP/UDP port name resolution (transport layer) 155
7.8. Checksums 156
7.8.1. Wireshark checksum validation 156
7.8.2. Checksum offloading 157
8. Statistics 159
8.1. Introduction 159
8.2. The "Summary" window 160
8.3. The "Protocol Hierarchy" window 163
8.4. Conversations 165
8.4.1. What is a Conversation? 165
8.4.2. The "Conversations" window 165
8.4.3. The protocol specific "Conversation List" windows 166
8.5. Endpoints 167
8.5.1. What is an Endpoint? 167
8.5.2. The "Endpoints" window 167
8.5.3. The protocol specific "Endpoint List" windows 168
8.6. The "IO Graphs" window 169
8.7. Compare two capture files 171
8.8. WLAN Traffic Statistics 172
8.9. LTE MAC Traffic Statistics 173
8.10. Service Response Time 174
8.10.1. The "Service Response Time DCE-RPC" window 174
8.11. The protocol specific statistics windows 176
9. Customizing Wireshark 178
9.1. Introduction 178
9.2. Start Wireshark from the command line 179
9.3. Packet colorization 185
9.4. Control Protocol dissection 189
9.4.1. The "Enabled Protocols" dialog box 189
9.4.2. User Specified Decodes 191
9.4.3. Show User Specified Decodes 192
9.5. Preferences 194
9.5.1. Interface Options 195
9.6. Configuration Profiles 197
9.7. User Table 200
9.8. Display Filter Macros 201
9.9. GeoIP Database Paths 202
9.10. Tektronix K12xx/15 RF5 protocols Table 203
9.11. PRES Users Context List 204
9.12. SCCP users Table 205
9.13. SMI (MIB and PIB) Modules 206
9.14. SMI (MIB and PIB) Paths 207
9.15. SNMP users Table 208
9.16. User DLTs protocol table 209
9.17. IKEv2 decryption table 210
10. Lua Support in Wireshark 212
10.1. Introduction 212
10.2. Example of Dissector written in Lua 213
10.3. Example of Listener written in Lua 214
10.4. Wireshark's Lua API Reference Manual 215
10.4.1. Saving capture files 215
10.4.2. Obtaining dissection data 217
10.4.3. GUI support 219
10.4.4. Post-dissection packet analysis 225
10.4.5. Obtaining packet information 226
10.4.6. Functions for writing dissectors 230
10.4.7. Adding information to the dissection tree 243
10.4.8. Functions for handling packet data 244

10.4.9. Utility Functions 250
A. Files and Folders 255
A.1. Capture Files 255
A.1.1. Libpcap File Contents 255
A.1.2. Not Saved in the Capture File 255
A.2. Configuration Files and Folders 257
A.3. Windows folders 262
A.3.1. Windows profiles 262
A.3.2. Windows Vista/XP/2000/NT roaming profiles 262
A.3.3. Windows temporary folder 263
B. Protocols and Protocol Fields 265
C. Wireshark Messages 266
C.1. Packet List Messages 266
C.1.1. [Malformed Packet] 266
C.1.2. [Packet size limited during capture] 266
C.2. Packet Details Messages 267
C.2.1. [Response in frame: 123] 267
C.2.2. [Request in frame: 123] 267
C.2.3. [Time from request: 0.123 seconds] 267
C.2.4. [Stream setup by PROTOCOL (frame 123)] 267
D. Related command line tools 269
D.1. Introduction 269
D.2. **tshark**: Terminal-based Wireshark 270
D.3. **tcpdump**: Capturing with tcpdump for viewing with Wireshark 271
D.4. **dumpcap**: Capturing with dumpcap for viewing with Wireshark 272
D.5. **capinfos**: Print information about capture files 273
D.6. **editcap**: Edit capture files 274
D.7. **mergecap**: Merging multiple capture files into one 277
D.8. **text2pcap**: Converting ASCII hexdumps to network captures 279
D.9. **idl2wrs**: Creating dissectors from CORBA IDL files 281
D.9.1. What is it? 281
D.9.2. Why do this? 281
D.9.3. How to use idl2wrs 281
D.9.4. TODO 283
D.9.5. Limitations 283
D.9.6. Notes 283
E. This Document's License (GPL) 285

Preface

1. Foreword

Wireshark is one of those programs that many network managers would love to be able to use, but they are often prevented from getting what they would like from Wireshark because of the lack of documentation.

This document is part of an effort by the Wireshark team to improve the usability of Wireshark.

We hope that you find it useful, and look forward to your comments.

2. Who should read this document?

The intended audience of this book is anyone using Wireshark.

This book will explain all the basics and also some of the advanced features that Wireshark provides. As Wireshark has become a very complex program since the early days, not every feature of Wireshark may be explained in this book.

This book is not intended to explain network sniffing in general and it will not provide details about specific network protocols. A lot of useful information regarding these topics can be found at the Wireshark Wiki at http://wiki.wireshark.org

By reading this book, you will learn how to install Wireshark, how to use the basic elements of the graphical user interface (such as the menu) and what's behind some of the advanced features that are not always obvious at first sight. It will hopefully guide you around some common problems that frequently appear for new (and sometimes even advanced) users of Wireshark.

3. Acknowledgements

The authors would like to thank the whole Wireshark team for their assistance. In particular, the authors would like to thank:

- Gerald Combs, for initiating the Wireshark project and funding to do this documentation.
- Guy Harris, for many helpful hints and a great deal of patience in reviewing this document.
- Gilbert Ramirez, for general encouragement and helpful hints along the way.

The authors would also like to thank the following people for their helpful feedback on this document:

- Pat Eyler, for his suggestions on improving the example on generating a backtrace.
- Martin Regner, for his various suggestions and corrections.
- Graeme Hewson, for a lot of grammatical corrections.

The authors would like to acknowledge those man page and README authors for the Wireshark project from who sections of this document borrow heavily:

- Scott Renfro from whose **mergecap** man page Section D.7, “**mergecap**: Merging multiple capture files into one ” is derived.
- Ashok Narayanan from whose **text2pcap** man page Section D.8, “**text2pcap**: Converting ASCII hexdumps to network captures ” is derived.
- Frank Singleton from whose `README.idl2wrs` Section D.9, “**idl2wrs**: Creating dissectors from CORBA IDL files ” is derived.

4. About this document

This book was originally developed by Richard Sharpe with funds provided from the Wireshark Fund. It was updated by Ed Warnicke and more recently redesigned and updated by Ulf Lamping.

It is written in DocBook/XML.

You will find some specially marked parts in this book:

This is a warning!

You should pay attention to a warning, as otherwise data loss might occur.

This is a note!

A note will point you to common mistakes and things that might not be obvious.

This is a tip!

Tips will be helpful for your everyday work using Wireshark.

5. Where to get the latest copy of this document?

The latest copy of this documentation can always be found at: http://www.wireshark.org/docs/.

6. Providing feedback about this document

Should you have any feedback about this document, please send it to the authors through wireshark-dev[AT]wireshark.org.

Chapter 1. Introduction

1.1. What is Wireshark?

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

You could think of a network packet analyzer as a measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed.

Wireshark is perhaps one of the best open source packet analyzers available today.

1.1.1. Some intended purposes

Here are some examples people use Wireshark for:

- network administrators use it to **troubleshoot network problems**
- network security engineers use it to **examine security problems**
- developers use it to **debug protocol implementations**
- people use it to **learn network protocol** internals

Beside these examples, Wireshark can be helpful in many other situations too.

1.1.2. Features

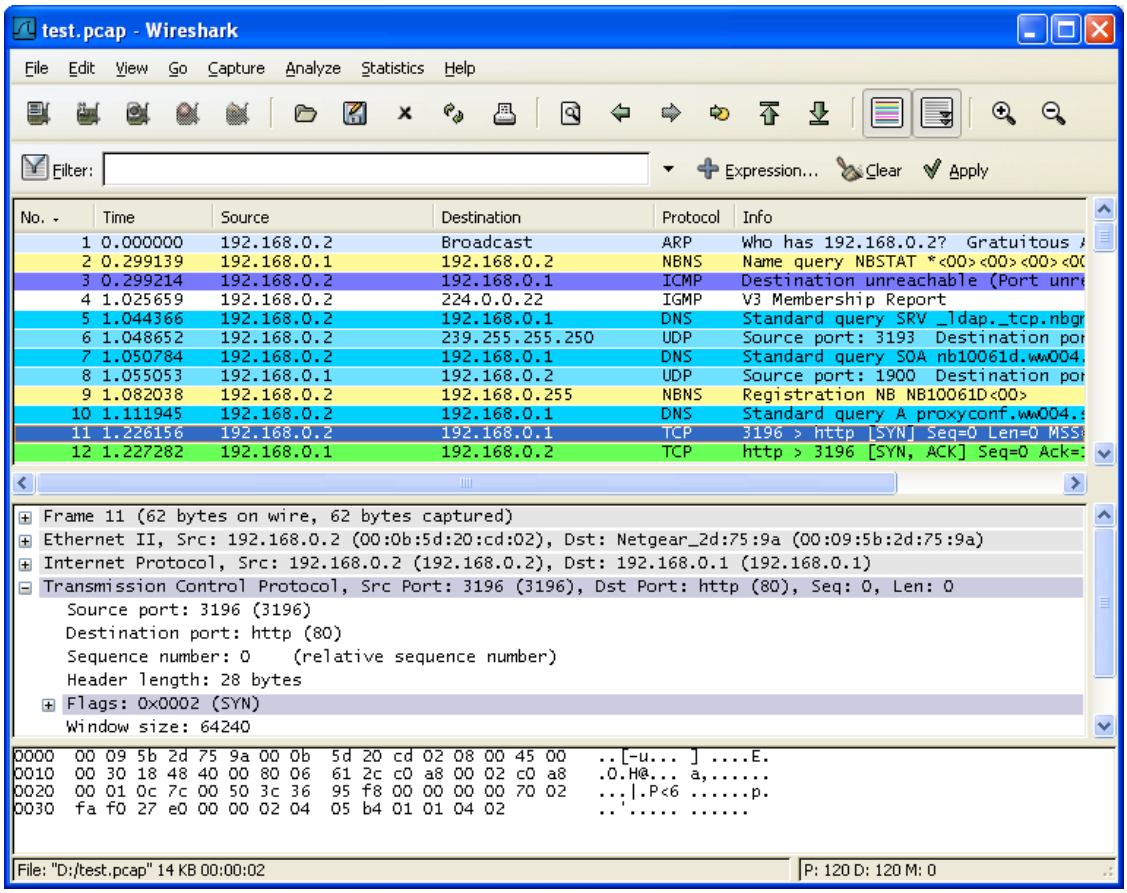
The following are some of the many features Wireshark provides:

- Available for **UNIX** and **Windows**.
- **Capture** live packet data from a network interface.
- Display packets with **very detailed protocol information**.
- **Open and Save** packet data captured.
- **Import and Export** packet data from and to a lot of other capture programs.
- **Filter packets** on many criteria.
- **Search** for packets on many criteria.
- **Colorize** packet display based on filters.
- Create various **statistics**.
- ... and **a lot more!**

However, to really appreciate its power, you have to start using it.

Figure 1.1, “ Wireshark captures packets and allows you to examine their content. ” shows Wireshark having captured some packets and waiting for you to examine them.

Figure 1.1. Wireshark captures packets and allows you to examine their content.


test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 192.168.0.2 Broadcast ARP Who has 192.168.0.2? Gratuitous
2 0.299139 192.168.0.1 192.168.0.2 NBNS Name query NBSTAT
3 0.299214 192.168.0.2 192.168.0.1 ICMP Destination unreachable (Port unr
4 1.025659 192.168.0.2 224.0.0.22 IGMP V3 Membership Report
5 1.044366 192.168.0.2 192.168.0.1 DNS Standard query SRV _ldap._tcp.nbg
6 1.048652 192.168.0.2 239.255.255.250 UDP Source port: 3193 Destination por
7 1.050784 192.168.0.2 192.168.0.1 DNS Standard query SOA nb100061d.ww004.
8 1.055053 192.168.0.1 192.168.0.2 UDP Source port: 1900 Destination por
9 1.082038 192.168.0.2 192.168.0.255 NBNS Registration NB NB100061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.ww004.
11 1.226156 192.168.0.2 192.168.0.1 TCP 3196 > http [SYN] Seq=0 Len=0 MSS
12 1.227282 192.168.0.1 192.168.0.2 TCP http > 3196 [SYN, ACK] Seq=0 Ack=
Frame 11 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: 3196 (3196)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x0002 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02
File: "D:/test.pcap" 14 KB 00:00:02
P: 120 D: 120 M: 0

1.1.3. Live capture from many different network media

Wireshark can capture traffic from many different network media types - and despite its name - including wireless LAN as well. Which media types are supported, depends on many things like the operating system you are using. An overview of the supported media types can be found at: http:// wiki.wireshark.org/CaptureSetup/NetworkMedia.

1.1.4. Import files from many other capture programs

Wireshark can open packets captured from a large number of other capture programs. For a list of input formats see Section 5.2.2, “Input File Formats”.

1.1.5. Export files for many other capture programs

Wireshark can save packets captured in a large number of formats of other capture programs. For a list of output formats see Section 5.3.2, “Output File Formats”.

1.1.6. Many protocol decoders

There are protocol decoders (or dissectors, as they are known in Wireshark) for a great many protocols: see Appendix B, *Protocols and Protocol Fields*.

1.1.7. Open Source Software

Wireshark is an open source software project, and is released under the GNU General Public License (GPL). You can freely use Wireshark on any number of computers you like, without worrying about license keys or fees or such. In addition, all source code is freely available under the GPL. Because of that, it is very easy for people to add new protocols to Wireshark, either as plugins, or built into the source, and they often do!

1.1.8. What Wireshark is not

Here are some things Wireshark does not provide:

- Wireshark isn't an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn't allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark doesn't send packets on the network or do other active things (except for name resolutions, but even that can be disabled).

1.2. System Requirements

What you'll need to get Wireshark up and running ...

1.2.1. General Remarks

- The values below are the minimum requirements and only "rules of thumb" for use on a moderately used network
- Working with a busy network can easily produce huge memory and disk space usage! For example: Capturing on a fully saturated 100MBit/s Ethernet will produce ~ 750MBytes/min! Having a fast processor, lots of memory and disk space is a good idea in that case.
- If Wireshark is running out of memory it crashes, see: http:/ / wiki.wireshark.org/ KnownBugs/ OutOfMemory for details and workarounds
- Wireshark won't benefit much from Multiprocessor/Hyperthread systems as time consuming tasks like filtering packets are single threaded. No rule is without exception: during an "Update list of packets in real time" capture, capturing traffic runs in one process and dissecting and displaying packets runs in another process - which should benefit from two processors.

1.2.2. Microsoft Windows

- Windows 2000, XP Home, XP Pro, XP Tablet PC, XP Media Center, Server 2003, Vista or Windows 2008 (XP Pro recommended)
- 32-bit Pentium or alike (recommended: 400MHz or greater), 64-bit processors in WoW64 emulation - see remarks below
- 128MB RAM system memory (recommended: 256MBytes or more)
- 75MB available disk space (plus size of user's capture files, e.g. 100MB extra)
- 800*600 (1280*1024 or higher recommended) resolution with at least 65536 (16bit) colors (256 colors should work if Wireshark is installed with the "legacy GTK1" selection of the Wireshark 1.0.x releases)
- A supported network card for capturing:
  - Ethernet: any card supported by Windows should do
  - WLAN: see the MicroLogix support list, no capturing of 802.11 headers and non-data frames
  - Other media: See http://wiki.wireshark.org/CaptureSetup/NetworkMedia

Remarks:

- Older Windows versions are no longer supported because of three reasons: None of the developers actively use those systems any longer which makes support difficult. The libraries Wireshark depends on (GTK, WinPCap, ...) are also dropping support for these systems. Microsoft also dropped support for these systems.

- Windows 95, 98 and ME will no longer work with Wireshark. The last known version to work was Ethereal 0.99.0 (which includes WinPcap 3.1). You can get it from http://ethereal.com/download.html. According to this bug report, you may need to install Ethereal 0.10.0 on some systems. BTW: Microsoft no longer supports 98/ME since July 11, 2006!
- Windows NT 4.0 will no longer work with Wireshark. The last known version to work was Wireshark 0.99.4 (which includes WinPcap 3.1), you still can get it from: http://prdownloads.sourceforge.net/wireshark/wireshark-setup-0.99.4.exe. BTW: Microsoft no longer supports NT 4.0 since December 31, 2005!
- Windows CE and the embedded (NT/XP) versions are not supported!
- 64-bit processors run Wireshark in 32 bit emulation (called WoW64), at least WinPcap 4.0 is required for that
- Multi monitor setups are supported but may behave a bit strangely

1.2.3. Unix / Linux

Wireshark currently runs on most UNIX platforms. The system requirements should be comparable to the Windows values listed above.

Binary packages are available for at least the following platforms:

- Apple Mac OS X
- Debian GNU/Linux
- FreeBSD
- Gentoo Linux
- HP-UX
- Mandriva Linux
- NetBSD
- OpenPKG
- Red Hat Fedora/Enterprise Linux
- rPath Linux
- Sun Solaris/i386
- Sun Solaris/Sparc

If a binary package is not available for your platform, you should download the source and try to build it. Please report your experiences to wireshark-dev[AT]wireshark.org .

1.3. Where to get Wireshark?

You can get the latest copy of the program from the Wireshark website: http://www.wireshark.org/download.html. The website allows you to choose from among several mirrors for downloading.

A new Wireshark version will typically become available every 4-8 months.

If you want to be notified about new Wireshark releases, you should subscribe to the wireshark-announce mailing list. You will find more details in Section 1.6.4, “Mailing Lists”.

1.4. A brief history of Wireshark

In late 1997, Gerald Combs needed a tool for tracking down networking problems and wanted to learn more about networking, so he started writing Ethereal (the former name of the Wireshark project) as a way to solve both problems.

Ethereal was initially released, after several pauses in development, in July 1998 as version 0.2.0. Within days, patches, bug reports, and words of encouragement started arriving, so Ethereal was on its way to success.

Not long after that, Gilbert Ramirez saw its potential and contributed a low-level dissector to it.

In October, 1998, Guy Harris of Network Appliance was looking for something better than tcpview, so he started applying patches and contributing dissectors to Ethereal.

In late 1998, Richard Sharpe, who was giving TCP/IP courses, saw its potential on such courses, and started looking at it to see if it supported the protocols he needed. While it didn't at that point, new protocols could be easily added. So he started contributing dissectors and contributing patches.

The list of people who have contributed to Ethereal has become very long since then, and almost all of them started with a protocol that they needed that Ethereal did not already handle. So they copied an existing dissector and contributed the code back to the team.

In 2006 the project moved house and re-emerged under a new name: Wireshark.

1.5. Development and maintenance of Wireshark

Wireshark was initially developed by Gerald Combs. Ongoing development and maintenance of Wireshark is handled by the Wireshark team, a loose group of individuals who fix bugs and provide new functionality.

There have also been a large number of people who have contributed protocol dissectors to Wireshark, and it is expected that this will continue. You can find a list of the people who have contributed code to Wireshark by checking the about dialog box of Wireshark, or at the authors page on the Wireshark web site.

Wireshark is an open source software project, and is released under the GNU General Public License (GPL). All source code is freely available under the GPL. You are welcome to modify Wireshark to suit your own needs, and it would be appreciated if you contribute your improvements back to the Wireshark team.

You gain three benefits by contributing your improvements back to the community:

- Other people who find your contributions useful will appreciate them, and you will know that you have helped people in the same way that the developers of Wireshark have helped people.
- The developers of Wireshark might improve your changes even more, as there's always room for improvement. Or they may implement some advanced things on top of your code, which can be useful for yourself too.
- The maintainers and developers of Wireshark will maintain your code as well, fixing it when API changes or other changes are made, and generally keeping it in tune with what is happening with Wireshark. So if Wireshark is updated (which is done often), you can get a new Wireshark version from the website and your changes will already be included without any effort for you.

The Wireshark source code and binary kits for some platforms are all available on the download page of the Wireshark website: http://www.wireshark.org/download.html.

1.6. Reporting problems and getting help

If you have problems, or need help with Wireshark, there are several places that may be of interest to you (well, besides this guide of course).

1.6.1. Website

You will find lots of useful information on the Wireshark homepage at http://www.wireshark.org.

1.6.2. Wiki

The Wireshark Wiki at http://wiki.wireshark.org provides a wide range of information related to Wireshark and packet capturing in general. You will find a lot of information not part of this user's guide. For example, there is an explanation how to capture on a switched network, an ongoing effort to build a protocol reference and a lot more.

And best of all, if you would like to contribute your knowledge on a specific topic (maybe a network protocol you know well), you can edit the wiki pages by simply using your web browser.

1.6.3. FAQ

The "Frequently Asked Questions" will list often asked questions and the corresponding answers.

Read the FAQ!

Before sending any mail to the mailing lists below, be sure to read the FAQ, as it will often answer the question(s) you might have. This will save yourself and others a lot of time (keep in mind that a lot of people are subscribed to the mailing lists).

You will find the FAQ inside Wireshark by clicking the menu item Help/Contents and selecting the FAQ page in the dialog shown.

An online version is available at the Wireshark website: http://www.wireshark.org/faq.html. You might prefer this online version, as it's typically more up to date and the HTML format is easier to use.

1.6.4. Mailing Lists

There are several mailing lists of specific Wireshark topics available:

wireshark-announce	This mailing list will inform you about new program releases, which usually appear about every 4-8 weeks.
wireshark-users	This list is for users of Wireshark. People post questions about building and using Wireshark, others (hopefully) provide answers.
wireshark-dev	This list is for Wireshark developers. If you want to start developing a protocol dissector, join this list.

You can subscribe to each of these lists from the Wireshark web site: http://www.wireshark.org. Simply select the **mailing lists** link on the left hand side of the site. The lists are archived at the Wireshark web site as well.

Tip!

You can search in the list archives to see if someone asked the same question some time before and maybe already got an answer. That way you don't have to wait until someone answers your question.

1.6.5. Reporting Problems

Note!

Before reporting any problems, please make sure you have installed the latest version of Wireshark.

When reporting problems with Wireshark, it is helpful if you supply the following information:

1. The version number of Wireshark and the dependent libraries linked with it, e.g. GTK+, etc. You can obtain this with the command **wireshark -v**.
2. Information about the platform you run Wireshark on.
3. A detailed description of your problem.
4. If you get an error/warning message, copy the text of that message (and also a few lines before and after it, if there are some), so others may find the place where things go wrong. Please don't give something like: "I get a warning while doing x" as this won't give a good idea where to look at.

Don't send large files!

Do not send large files (>100KB) to the mailing lists, just place a note that further data is available on request. Large files will only annoy a lot of people on the list who are not interested in your specific problem. If required, you will be asked for further data by the persons who really can help you.

Don't send confidential information!

If you send captured data to the mailing lists, be sure they don't contain any sensitive or confidential information like passwords or such.

1.6.6. Reporting Crashes on UNIX/Linux platforms

When reporting crashes with Wireshark, it is helpful if you supply the traceback information (besides the information mentioned in "Reporting Problems").

You can obtain this traceback information with the following commands:

```
$ gdb `whereis wireshark | cut -f2 -d: | cut -d' ' -f2` core >& bt.txt
backtrace
^D
$
```

Note

Type the characters in the first line verbatim! Those are back-tics there!

Note

backtrace is a **gdb** command. You should enter it verbatim after the first line shown above, but it will not be echoed. The ^D (Control-D, that is, press the Control key and the D key together) will cause **gdb** to exit. This will leave you with a file called `bt.txt` in the current directory. Include the file with your bug report.

Note

If you do not have **gdb** available, you will have to check out your operating system's debugger.

You should mail the traceback to the wireshark-dev[AT]wireshark.org mailing list.

1.6.7. Reporting Crashes on Windows platforms

The Windows distributions don't contain the symbol files (.pdb), because they are very large. For this reason it's not possible to create a meaningful backtrace file from it. You should report your crash just like other problems, using the mechanism described above.

Chapter 2. Building and Installing Wireshark

2.1. Introduction

As with all things, there must be a beginning, and so it is with Wireshark. To use Wireshark, you must:

- Obtain a binary package for your operating system, or
- Obtain the source and build Wireshark for your operating system.

Currently, only two or three Linux distributions ship Wireshark, and they are commonly shipping an out-of-date version. No other versions of UNIX ship Wireshark so far, and Microsoft does not ship it with any version of Windows. For that reason, you will need to know where to get the latest version of Wireshark and how to install it.

This chapter shows you how to obtain source and binary packages, and how to build Wireshark from source, should you choose to do so.

The following are the general steps you would use:

1. Download the relevant package for your needs, e.g. source or binary distribution.
2. Build the source into a binary, if you have downloaded the source.

   This may involve building and/or installing other necessary packages.
3. Install the binaries into their final destinations.

2.2. Obtaining the source and binary distributions

You can obtain both source and binary distributions from the Wireshark web site: http://www.wireshark.org. Simply select the download link, and then select either the source package or binary package of your choice from the mirror site closest to you.

Download all required files!

In general, unless you have already downloaded Wireshark before, you will most likely need to download several source packages if you are building Wireshark from source. This is covered in more detail below.

Once you have downloaded the relevant files, you can go on to the next step.

Note!

While you will find a number of binary packages available on the Wireshark web site, you might not find one for your platform, and they often tend to be several versions behind the current released version, as they are contributed by people who have the platforms they are built for.

For this reason, you might want to pull down the source distribution and build it, as the process is relatively simple.

2.3. Before you build Wireshark under UNIX

Before you build Wireshark from sources, or install a binary package, you must ensure that you have the following other packages installed:

- GTK+, The GIMP Tool Kit.

  You will also need Glib. Both can be obtained from www.gtk.org

- libpcap, the packet capture software that Wireshark uses.

  You can obtain libpcap from www.tcpdump.org

Depending on your system, you may be able to install these from binaries, e.g. RPMs, or you may need to obtain them in source code form and build them.

If you have downloaded the source for GTK+, the instructions shown in Example 2.1, "Building GTK+ from source" may provide some help in building it:

Example 2.1. Building GTK+ from source

```
gzip -dc gtk+-1.2.10.tar.gz | tar xvf -
<much output removed>
cd gtk+-1.2.10
./configure
<much output removed>
make
<much output removed>
make install
<much output removed>
```

Note!

You may need to change the version number of gtk+ in Example 2.1, "Building GTK+ from source" to match the version of GTK+ you have downloaded. The directory you change to will change if the version of GTK+ changes, and in all cases, **tar xvf -** will show you the name of the directory you should change to.

Note!

If you use Linux, or have GNU **tar** installed, you can use **tar zxvf gtk+-1.2.10.tar.gz**. It is also possible to use **gunzip -c** or **gzcat** rather than **gzip -dc** on many UNIX systems.

Note!

If you downloaded gtk+ or any other tar file using Windows, you may find your file called gtk+-1_2_8_tar.gz.

You should consult the GTK+ web site if any errors occur in carrying out the instructions in Example 2.1, "Building GTK+ from source".

If you have downloaded the source to libpcap, the general instructions shown in Example 2.2, "Building and installing libpcap" will assist in building it. Also, if your operating system does not support **tcpdump**, you might also want to download it from the tcpdump web site and install it.

Example 2.2. Building and installing libpcap

```
gzip -dc libpcap-0.9.4.tar.Z | tar xvf -
<much output removed>
cd libpcap-0.9.4
./configure
<much output removed>
make
<much output removed>
make install
<much output removed>
```

Note!

The directory you should change to will depend on the version of libpcap you have downloaded. In all cases, **tar xvf -** will show you the name of the directory that has been unpacked.

Under Red Hat 6.x and beyond (and distributions based on it, like Mandrake) you can simply install each of the packages you need from RPMs. Most Linux systems will install GTK+ and GLib in any case, however you will probably need to install the devel versions of each of these packages. The commands shown in Example 2.3, " Installing required RPMs under Red Hat Linux 6.2 and beyond " will install all the needed RPMs if they are not already installed.

Example 2.3. Installing required RPMs under Red Hat Linux 6.2 and beyond

```
cd /mnt/cdrom/RedHat/RPMS
rpm -ivh glib-1.2.6-3.i386.rpm
rpm -ivh glib-devel-1.2.6-3.i386.rpm
rpm -ivh gtk+-1.2.6-7.i386.rpm
rpm -ivh gtk+-devel-1.2.6-7.i386.rpm
rpm -ivh libpcap-0.4-19.i386.rpm
```

Note

If you are using a version of Red Hat later than 6.2, the required RPMs have most likely changed. Simply use the correct RPMs from your distribution.

Under Debian you can install Wireshark using aptitude. aptitude will handle any dependency issues for you. Example 2.4, "Installing debs under Debian" shows how to do this.

Example 2.4. Installing debs under Debian

```
aptitude install wireshark-dev
```

2.4. Building Wireshark from source under UNIX

Use the following general steps if you are building Wireshark from source under a UNIX operating system:

1. Unpack the source from its **gzip**'d **tar** file. If you are using Linux, or your version of UNIX uses GNU **tar**, you can use the following command:

   ```
   tar zxvf wireshark-1.0.0-tar.gz
   ```

   For other versions of UNIX, you will want to use the following commands:

   ```
   gzip -d wireshark-1.0.0-tar.gz
   tar xvf wireshark-1.0.0-tar
   ```

   Note!

   The pipeline **gzip -dc wireshark-1.0.0-tar.gz | tar xvf -** will work here as well.

   Note!

   If you have downloaded the Wireshark tarball under Windows, you may find that your browser has created a file with underscores rather than periods in its file name.

2. Change directory to the Wireshark source directory.

3. Configure your source so it will build correctly for your version of UNIX. You can do this with the following command:

   ```
   ./configure
   ```

   If this step fails, you will have to rectify the problems and rerun **configure**. Troubleshooting hints are provided in Section 2.6, “Troubleshooting during the install on Unix”.

4. Build the sources into a binary, with the **make** command. For example:

   ```
   make
   ```

5. Install the software in its final destination, using the command:

   ```
   make install
   ```

Once you have installed Wireshark with **make install** above, you should be able to run it by entering

wireshark.

2.5. Installing the binaries under UNIX

In general, installing the binary under your version of UNIX will be specific to the installation methods used with your version of UNIX. For example, under AIX, you would use **smit** to install the Wireshark binary package, while under Tru64 UNIX (formerly Digital UNIX) you would use **setld**.

2.5.1. Installing from rpm's under Red Hat and alike

Use the following command to install the Wireshark RPM that you have downloaded from the Wireshark web site:

```
rpm -ivh wireshark-1.0.0.i386.rpm
```

If the above step fails because of missing dependencies, install the dependencies first, and then retry the step above. See Example 2.3, “ Installing required RPMs under Red Hat Linux 6.2 and beyond ” for information on what RPMs you will need to have installed.

2.5.2. Installing from deb's under Debian

Use the following command to install Wireshark under Debian:

```
aptitude install wireshark
```

aptitude should take care of all of the dependency issues for you.

2.5.3. Installing from portage under Gentoo Linux

Use the following command to install Wireshark under Gentoo Linux with all of the extra features:

```
USE="adns gtk ipv6 portaudio snmp ssl kerberos threads selinux" emerge wireshark
```

2.5.4. Installing from packages under FreeBSD

Use the following command to install Wireshark under FreeBSD:

```
pkg_add -r wireshark
```

pkg_add should take care of all of the dependency issues for you.

2.6. Troubleshooting during the install on Unix

A number of errors can occur during the installation process. Some hints on solving these are provided here.

If the **configure** stage fails, you will need to find out why. You can check the file `config.log` in the source directory to find out what failed. The last few lines of this file should help in determining the problem.

The standard problems are that you do not have GTK+ on your system, or you do not have a recent enough version of GTK+. The **configure** will also fail if you do not have libpcap (at least the required include files) on your system.

Another common problem is for the final compile and link stage to terminate with a complaint of: Output too long. This is likely to be caused by an antiquated **sed** (such as the one shipped with Solaris). Since **sed** is used by the **libtool** script to construct the final link command, this leads to mysterious problems. This can be resolved by downloading a recent version of sed from http://directory.fsf.org/GNU/sed.html.

If you cannot determine what the problems are, send mail to the **wireshark-dev** mailing list explaining your problem, and including the output from `config.log` and anything else you think is relevant, like a trace of the **make** stage.

2.7. Building from source under Windows

It is recommended to use the binary installer for Windows, until you want to start developing Wireshark on the Windows platform.

For further information how to build Wireshark for Windows from the sources, have a look at the Development Wiki: http://wiki.wireshark.org/Development for the latest available development documentation.

2.8. Installing Wireshark under Windows

In this section we explore installing Wireshark under Windows from the binary packages.

2.8.1. Install Wireshark

You may acquire a binary installer of Wireshark named something like: `wireshark-setup-x.y.z.exe`. The Wireshark installer includes WinPcap, so you don't need to download and install two separate packages.

Simply download the Wireshark installer from: http://www.wireshark.org/download.html and execute it. Beside the usual installer options like where to install the program, there are several optional components.

Tip: Just keep the defaults!

If you are unsure which settings to select, just keep the defaults.

2.8.1.1. "Choose Components" page

Wireshark

- **Wireshark GTK** - Wireshark is a GUI network protocol analyzer.
- **GTK MS Windows Engine** - GTK MS Windows Engine (native Win32 look and feel, recommended).

TShark - TShark is a command-line based network protocol analyzer.

Plugins / Extensions (for the Wireshark and TShark dissection engines):

- **Dissector Plugins** - Plugins with some extended dissections.
- **Tree Statistics Plugins** - Plugins with some extended statistics.
- **Mate - Meta Analysis and Tracing Engine (experimental)** - user configurable extension(s) of the display filter engine, see http://wiki.wireshark.org/Mate for details.
- **SNMP MIBs** - SNMP MIBs for a more detailed SNMP dissection.

Tools (additional command line tools to work with capture files):

- **Editcap** - Editcap is a program that reads a capture file and writes some or all of the packets into another capture file.
- **Text2Pcap** - Text2pcap is a program that reads in an ASCII hex dump and writes the data into a libpcap-style capture file.
- **Mergecap** - Mergecap is a program that combines multiple saved capture files into a single output file.
- **Capinfos** - Capinfos is a program that provides information on capture files.

User's Guide - Local installation of the User's Guide. The Help buttons on most dialogs will require an internet connection to show help pages if the User's Guide is not installed locally.

2.8.1.2. "Additional Tasks" page

- **Start Menu Shortcuts** - add some start menu shortcuts.
- **Desktop Icon** - add a Wireshark icon to the desktop.
- **Quick Launch Icon** - add a Wireshark icon to the Explorer quick launch toolbar.
- **Associate file extensions to Wireshark** - Associate standard network trace files to Wireshark.

2.8.1.3. "Install WinPcap?" page

The Wireshark installer contains the latest released WinPcap installer.

If you don't have WinPcap installed, you won't be able to capture live network traffic, but you will still be able to open saved capture files.

- **Currently installed WinPcap version** - the Wireshark installer detects the currently installed WinPcap version.
- **Install WinPcap x.x** - if the currently installed version is older than the one which comes with the Wireshark installer (or WinPcap is not installed at all), this will be selected by default.
- **Start WinPcap service "NPF" at startup** - so users without administrative privileges can capture.

More WinPcap info:

- Wireshark related: http://wiki.wireshark.org/WinPcap
- General WinPcap info: http://www.winpcap.org

2.8.1.4. Command line options

You can simply start the Wireshark installer without any command line parameters, it will show you the usual interactive installer.

For special cases, there are some command line parameters available:

- **/NCRC** disables the CRC check
- **/S** runs the installer or uninstaller silently with default values. Please note: The silent installer won't install WinPCap!
- **/desktopicon** installation of the desktop icon, **=yes** - force installation, **=no** - don't install, otherwise use defaults / user settings. This option can be useful for a silent installer.
- **/quicklaunchicon** installation of the quick launch icon, **=yes** - force installation, **=no** - don't install,

otherwise use defaults / user settings.

- **/D** sets the default installation directory ($INSTDIR), overriding InstallDir and InstallDirRegKey. It must be the last parameter used in the command line and must not contain any quotes, even if the path contains spaces.

Example:

```
wireshark-setup-1.0.0.exe /NCRC /S /desktopicon=yes
  /quicklaunchicon=no /D=C:\Program Files\Foo
```

2.8.2. Manual WinPcap Installation

Note!

As mentioned above, the Wireshark installer takes care of the installation of WinPcap, so usually you don't have to worry about WinPcap at all!

The following is only necessary if you want to try a different version than the one included in the Wireshark installer, e.g. because a new WinPcap (beta) version was released.

Additional WinPcap versions (including newer alpha or beta releases) can be downloaded from the following locations:

- The main WinPcap site: http://www.winpcap.org
- The Wiretapped.net mirror: http://www.mirrors.wiretapped.net/security/packet-capture/winpcap

At the download page you will find a single installer exe called something like "auto-installer", which can be installed under various Windows systems, including NT4.0/2000/XP/Vista.

2.8.3. Update Wireshark

From time to time you may want to update your installed Wireshark to a more recent version. If you join Wireshark's announce mailing list, you will be informed about new Wireshark versions, see Section 1.6.4, “Mailing Lists” for details how to subscribe to this list.

New versions of Wireshark usually become available every 4 to 8 months. Updating Wireshark is done the same way as installing it, you simply download and start the installer exe. A reboot is usually not required and all your personal settings remain unchanged.

2.8.4. Update WinPcap

New versions of WinPcap are less frequently available, maybe only once in a year. You will find WinPcap update instructions where you can download new WinPcap versions. Usually you have to reboot the machine after installing a new WinPcap version.

Warning!

If you have an older version of WinPcap installed, you must uninstall it before installing the current version. Recent versions of the WinPcap installer will take care of this.

2.8.5. Uninstall Wireshark

You can uninstall Wireshark the usual way, using the "Add or Remove Programs" option inside the Control Panel. Select the "Wireshark" entry to start the uninstallation procedure.

The Wireshark uninstaller will provide several options as to which things are to be uninstalled; the default is to remove the core components but keep the personal settings, WinPcap and alike.

WinPcap won't be uninstalled by default, as other programs than Wireshark may use it as well.

2.8.6. Uninstall WinPcap

You can uninstall WinPcap independently of Wireshark, using the "WinPcap" entry in the "Add or Remove Programs" of the Control Panel.

Note!

After uninstallation of WinPcap you can't capture anything with Wireshark.

It might be a good idea to reboot Windows afterwards.

Chapter 3. User Interface

3.1. Introduction

By now you have installed Wireshark and are most likely keen to get started capturing your first packets. In the next chapters we will explore:

- How the Wireshark user interface works
- How to capture packets in Wireshark
- How to view packets in Wireshark
- How to filter packets in Wireshark
- ... and many other things!

3.2. Start Wireshark

You can start Wireshark from your shell or window manager.

Tip!

When starting Wireshark it's possible to specify optional settings using the command line. See Section 9.2, “Start Wireshark from the command line” for details.

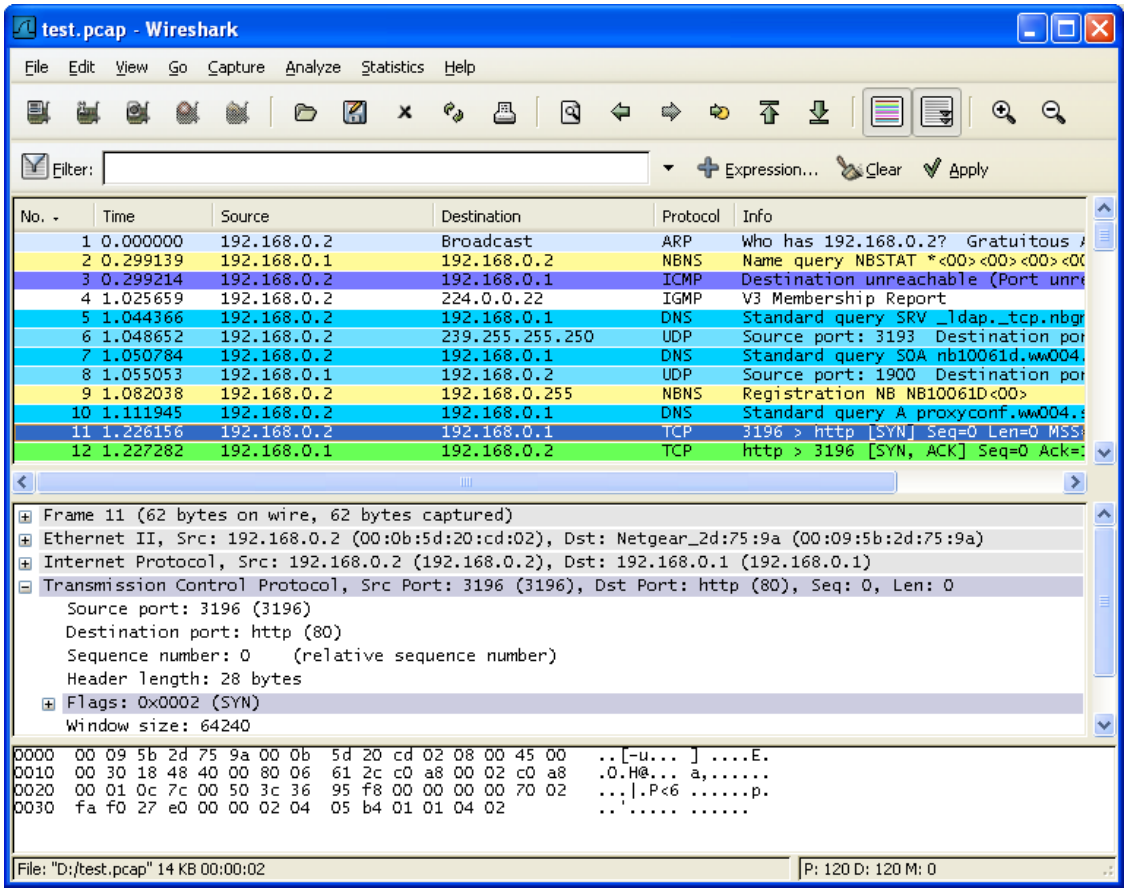
Note!

In the following chapters, a lot of screenshots from Wireshark will be shown. As Wireshark runs on many different platforms and there are different versions of the underlying GUI toolkit (GTK 1.x / 2.x) used, your screen might look different from the provided screenshots. But as there are no real differences in functionality, these screenshots should still be well understandable.

3.3. The Main window

Let's look at Wireshark's user interface. Figure 3.1, “The Main window” shows Wireshark as you would usually see it after some packets are captured or loaded (how to do this will be described later).

Figure 3.1. The Main window


test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 192.168.0.2 Broadcast ARP Who has 192.168.0.2? Gratuitous
2 0.299139 192.168.0.1 192.168.0.2 NBNS Name query NBSTAT *<00><00><00><0
3 0.299214 192.168.0.2 192.168.0.1 ICMP Destination unreachable (Port unr
4 1.025659 192.168.0.2 224.0.0.22 IGMP V3 Membership Report
5 1.044366 192.168.0.2 192.168.0.1 DNS Standard query SRV _ldap._tcp.nbgr
6 1.048652 192.168.0.2 239.255.255.250 UDP Source port: 3193 Destination por
7 1.050784 192.168.0.2 192.168.0.1 DNS Standard query SOA nb10061d.ww004.
8 1.055053 192.168.0.1 192.168.0.2 UDP Source port: 1900 Destination por
9 1.082038 192.168.0.2 192.168.0.255 NBNS Registration NB NB10061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.ww004.
11 1.226156 192.168.0.2 192.168.0.1 TCP 3196 > http [SYN] Seq=0 Len=0 MSS
12 1.227282 192.168.0.1 192.168.0.2 TCP http > 3196 [SYN, ACK] Seq=0 Ack=
Frame 11 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: 3196 (3196)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x0002 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00 ..[-u...] ....E.
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8 .0.H@... a,.....
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02 ...|.P<6 ......p.
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02 ..'..... ......
File: "D:/test.pcap" 14 KB 00:00:02
P: 120 D: 120 M: 0

Wireshark's main window consists of parts that are commonly known from many other GUI programs.

1. The *menu* (see Section 3.4, “The Menu”) is used to start actions.

2. The *main toolbar* (see Section 3.14, “The "Main" toolbar”) provides quick access to frequently used items from the menu.

3. The *filter toolbar* (see Section 3.15, “The "Filter" toolbar”) provides a way to directly manipulate the currently used display filter (see Section 6.3, “Filtering packets while viewing”).

4. The *packet list pane* (see Section 3.16, “The "Packet List" pane”) displays a summary of each packet captured. By clicking on packets in this pane you control what is displayed in the other two panes.

5. The *packet details pane* (see Section 3.17, “The "Packet Details" pane”) displays the packet selected in the packet list pane in more detail.

6. The *packet bytes pane* (see Section 3.18, “The "Packet Bytes" pane”) displays the data from the packet selected in the packet list pane, and highlights the field selected in the packet details pane.

7. The *statusbar* (see Section 3.19, “The Statusbar”) shows some detailed information about the current program state and the captured data.

Tip!

The layout of the main window can be customized by changing preference settings. See Section 9.5, “Preferences” for details!

3.3.1. Main Window Navigation

Packet list and detail navigation can be done entirely from the keyboard. Table 3.1, “Keyboard Navigation” shows a list of keystrokes that will let you quickly move around a capture file. See Table 3.5, “Go menu items” for additional navigation keystrokes.

Table 3.1. Keyboard Navigation

Accelerator	Description
Tab, Shift+Tab	Move between screen elements, e.g. from the toolbars to the packet list to the packet detail.
Down	Move to the next packet or detail item.
Up	Move to the previous packet or detail item.
Ctrl+Down, F8	Move to the next packet, even if the packet list isn't focused.
Ctrl+Up, F7	Move to the previous packet, even if the packet list isn't focused.
Left	In the packet detail, closes the selected tree item. If it's already closed, jumps to the parent node.
Right	In the packet detail, opens the selected tree item.
Shift+Right	In the packet detail, opens the selected tree item and all of its subtrees.
Ctrl+Right	In the packet detail, opens all tree items.
Ctrl+Left	In the packet detail, closes all tree items.
Backspace	In the packet detail, jumps to the parent node.
Return, Enter	In the packet detail, toggles the selected tree item.

Additionally, typing anywhere in the main window will start filling in a display filter.

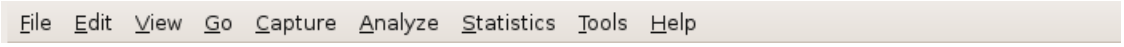
3.4. The Menu

The Wireshark menu sits on top of the Wireshark window. An example is shown in Figure 3.2, “The Menu”.

Note!

Menu items will be greyed out if the corresponding feature isn't available. For example, you cannot save a capture file if you didn't capture or load any data before.

Figure 3.2. The Menu



It contains the following items:

File This menu contains items to open and merge capture files, save / print / export capture files in whole or in part, and to quit from Wireshark. See Section 3.5, “The "File" menu”.

Edit This menu contains items to find a packet, time reference or mark one or more packets, handle configuration profiles, and set your preferences; (cut, copy, and paste are not presently implemented). See Section 3.6, “The "Edit" menu”.

View This menu controls the display of the captured data, including colorization of packets, zooming the font, showing a packet in a separate window, expanding and collapsing trees in packet details, ... See Section 3.7, “The "View" menu”.

Go This menu contains items to go to a specific packet. See Section 3.8, “The "Go" menu”.

Capture This menu allows you to start and stop captures and to edit capture filters. See Section 3.9, “The "Capture" menu”.

Analyze This menu contains items to manipulate display filters, enable or disable the dissection of protocols, configure user specified decodes and follow a TCP stream. See Section 3.10, “The "Analyze" menu”.

Statistics This menu contains items to display various statistic windows, including a summary of the packets that have been captured, display protocol hierarchy statistics and much more. See Section 3.11, “The "Statistics" menu”.

Tools This menu contains various tools available in Wireshark, such as creating Firewall ACL Rules. See Section 3.12, “The "Tools" menu”.

Help This menu contains items to help the user, e.g. access to some basic help, a list of the supported protocols, manual pages, online access to some of the webpages, and the usual about dialog. See Section 3.13, “The "Help" menu”.

Each of these menu items is described in more detail in the sections that follow.

Tip!

You can access menu items directly or by pressing the corresponding accelerator keys which are shown at the right side of the menu. For example, you can press the Control (or Strg in German) and the K keys together to open the capture dialog.

3.5. The "File" menu

The Wireshark file menu contains the fields shown in Table 3.2, “File menu items”.

Figure 3.3. The "File" Menu

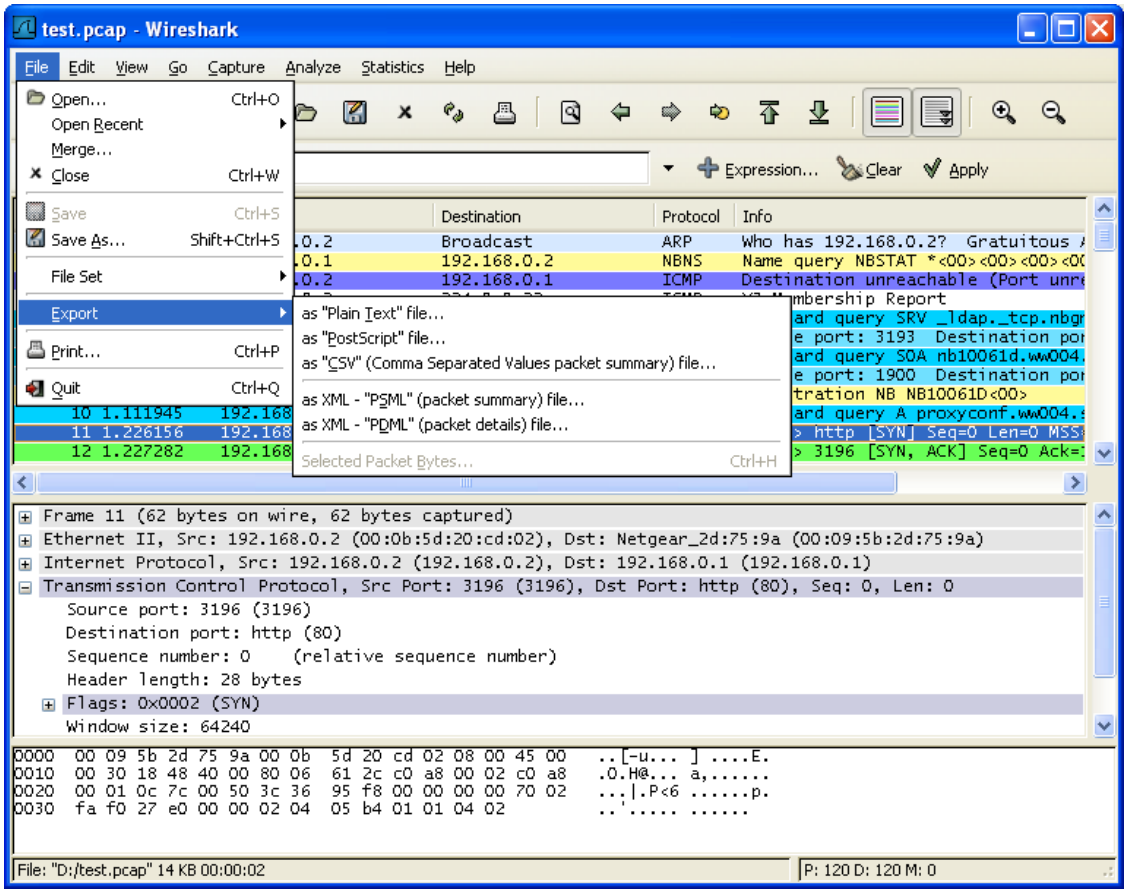

test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Open... Ctrl+O
Open Recent
Merge...
Close Ctrl+W
Save Ctrl+S
Save As... Shift+Ctrl+S
File Set
Export
Print... Ctrl+P
Quit Ctrl+Q
as "Plain Text" file...
as "PostScript" file...
as "CSV" (Comma Separated Values packet summary) file...
as XML - "PSML" (packet summary) file...
as XML - "PDML" (packet details) file...
Selected Packet Bytes... Ctrl+H
Expression... Clear Apply
Destination Protocol Info
Broadcast ARP Who has 192.168.0.2? Gratuitous
192.168.0.2 NBNS Name query NBSTAT
192.168.0.1 ICMP Destination unreachable (Port
Frame 11 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: 3196 (3196)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x0002 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02
File: "D:/test.pcap" 14 KB 00:00:02
P: 120 D: 120 M: 0

Table 3.2. File menu items

Menu Item	Accelerator	Description
Open...	Ctrl+O	This menu item brings up the file open dialog box that allows you to load a capture file for viewing. It is discussed in more detail in Section 5.2.1, “The "Open Capture File" dialog box”.
Open Recent		This menu item shows a submenu containing the recently opened capture files. Clicking on one of the submenu items will open the corresponding capture file directly.
Merge...		This menu item brings up the merge file dialog box that allows you to merge a capture file into the currently loaded one. It is discussed in more detail in Section 5.4, “Merging capture files”.

Menu Item	Accelerator	Description
Close	Ctrl+W	This menu item closes the current capture. If you haven't saved the capture, you will be asked to do so first (this can be disabled by a preference setting).

Save	Ctrl+S	This menu item saves the current capture. If you have not set a default capture file name (perhaps with the -w <capfile> option), Wireshark pops up the Save Capture File As dialog box (which is discussed further in Section 5.3.1, "The "Save Capture File As" dialog box"). **Note!** If you have already saved the current capture, this menu item will be greyed out. **Note!** You cannot save a live capture while the capture is in progress. You must stop the capture in order to save.
Save As...	Shift+Ctrl+S	This menu item allows you to save the current capture file to whatever file you would like. It pops up the Save Capture File As dialog box (which is discussed further in Section 5.3.1, "The "Save Capture File As" dialog box").

File Set > List Files		This menu item allows you to show a list of files in a file set. It pops up the Wireshark List File Set dialog box (which is discussed further in Section 5.5, "File Sets").
File Set > Next File		If the currently loaded file is part of a file set, jump to the next file in the set. If it isn't part of a file set or just the last file in that set, this item is greyed out.
File Set > Previous File		If the currently loaded file is part of a file set, jump to the previous file in the set. If it isn't part of a file set or just the first file in that set, this item is greyed out.

Export > as "Plain Text" file...		This menu item allows you to export all (or some) of the packets in the capture file to a plain ASCII text file. It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.1, "The "Export as Plain Text File" dialog box").
Export > as "PostScript" file...		This menu item allows you to export all (or some) of the packets in the capture file to a PostScript file. It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.2, "The "Export as PostScript File" dialog box").

Menu Item	Accelerator	Description
Export > as "CSV" (Comma Separated Values packet summary) file...		This menu item allows you to export all (or some) of the packet summaries in the capture file to a .csv file (e.g. used by spreadsheet programs). It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.3, “The "Export as CSV (Comma Separated Values) File" dialog box”).
Export > as "C Arrays" (packet bytes) file...		This menu item allows you to export all (or some) of the packet bytes in the capture file to a .c file so you can import the stream data into your own C program. It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.4, “The "Export as C Arrays (packet bytes) file" dialog box”).

Export > as "PSML" file...		This menu item allows you to export all (or some) of the packets in the capture file to a PSML (packet summary markup language) XML file. It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.5, “The "Export as PSML File" dialog box”).
Export > as "PDML" file...		This menu item allows you to export all (or some) of the packets in the capture file to a PDML (packet details markup language) XML file. It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.6, “The "Export as PDML File" dialog box”).

Export > Selected Packet Bytes...	Ctrl+H	This menu item allows you to export the currently selected bytes in the packet bytes pane to a binary file. It pops up the Wireshark Export dialog box (which is discussed further in Section 5.6.7, “The "Export selected packet bytes" dialog box”)
Objects > HTTP		This menu item allows you to export all or some of the captured HTTP objects into local files. It pops up the Wireshark HTTP object list (which is discussed further in Section 5.6.8, “The "Export Objects" dialog box”)

Print...	Ctrl+P	This menu item allows you to print all (or some) of the packets in the capture file. It pops up the Wireshark Print dialog box (which is discussed further in Section 5.7, “Printing packets”).

Quit	Ctrl+Q	This menu item allows you to quit from Wireshark. Wireshark will ask to save your capture file if you haven't previously saved it (this can be disabled by a preference setting).

3.6. The "Edit" menu

The Wireshark Edit menu contains the fields shown in Table 3.3, “Edit menu items”.

Figure 3.4. The "Edit" Menu

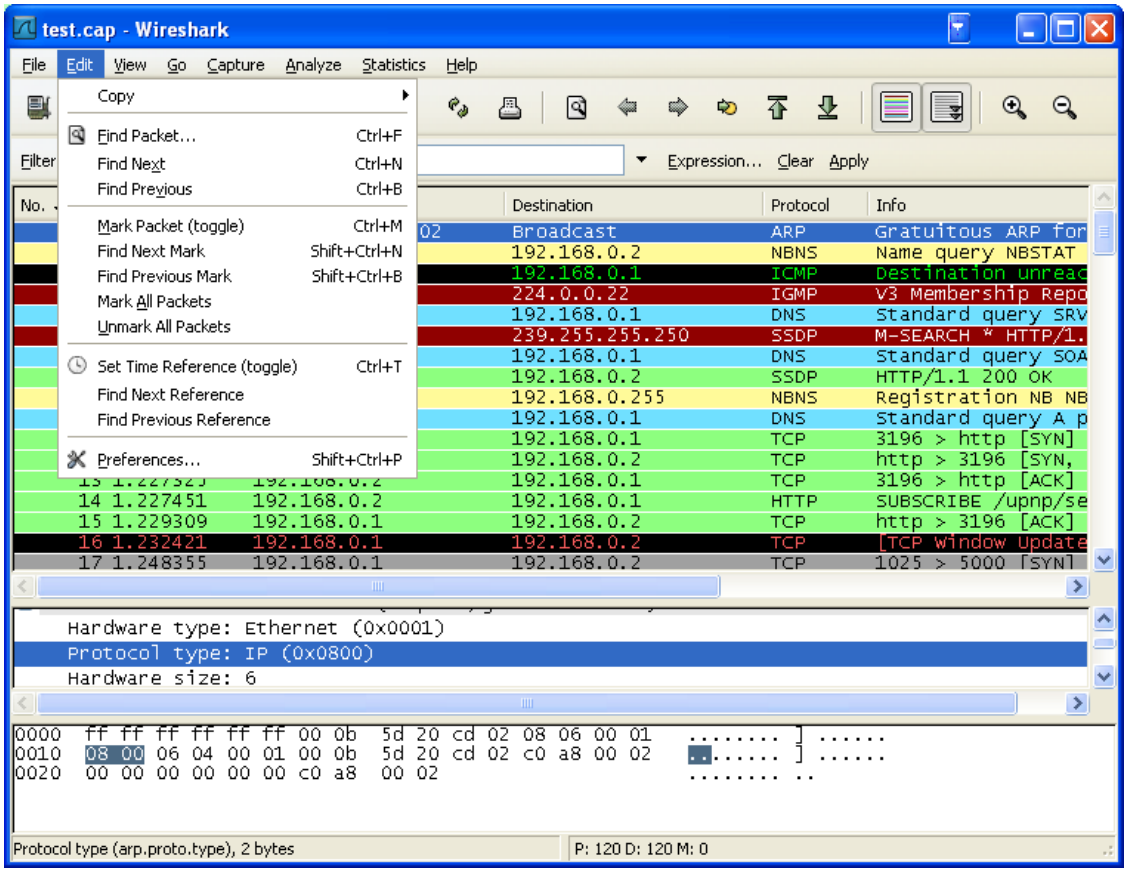

test.cap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Copy
Find Packet... Ctrl+F
Find Next Ctrl+N
Find Previous Ctrl+B
Mark Packet (toggle) Ctrl+M
Find Next Mark Shift+Ctrl+N
Find Previous Mark Shift+Ctrl+B
Mark All Packets
Unmark All Packets
Set Time Reference (toggle) Ctrl+T
Find Next Reference
Find Previous Reference
Preferences... Shift+Ctrl+P
Filter
Expression... Clear Apply
No. Destination Protocol Info
Broadcast ARP Gratuitous ARP for
192.168.0.2 NBNS Name query NBSTAT
192.168.0.1 ICMP Destination unreac
224.0.0.22 IGMP V3 Membership Repo
192.168.0.1 DNS Standard query SRV
239.255.255.250 SSDP M-SEARCH * HTTP/1.
192.168.0.1 DNS Standard query SOA
192.168.0.2 SSDP HTTP/1.1 200 OK
192.168.0.255 NBNS Registration NB NB
192.168.0.1 DNS Standard query A p
192.168.0.1 TCP 3196 > http [SYN]
192.168.0.2 TCP http > 3196 [SYN,
192.168.0.1 TCP 3196 > http [ACK]
14 1.227451 192.168.0.2 192.168.0.1 HTTP SUBSCRIBE /upnp/se
15 1.229309 192.168.0.1 192.168.0.2 TCP http > 3196 [ACK]
16 1.232421 192.168.0.1 192.168.0.2 TCP [TCP Window Update
17 1.248355 192.168.0.1 192.168.0.2 TCP 1025 > 5000 [SYN]
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
0000 ff ff ff ff ff ff 00 0b 5d 20 cd 02 08 06 00 01
0010 08 00 06 04 00 01 00 0b 5d 20 cd 02 c0 a8 00 02
0020 00 00 00 00 00 00 c0 a8 00 02
Protocol type (arp.proto.type), 2 bytes
P: 120 D: 120 M: 0

Table 3.3. Edit menu items

Menu Item	Accelerator	Description
Copy > Description	Shift+Ctrl+D	This menu item will copy the description of the selected item in the detail view to the clipboard.
Copy > Fieldname	Shift+Ctrl+F	This menu item will copy the fieldname of the selected item in the detail view to the clipboard.
Copy > Value	Shift+Ctrl+V	This menu item will copy the value of the selected item in the detail view to the clipboard.
Copy > As Filter	Shift+Ctrl+C	This menu item will use the selected item in the detail view to create a display filter. This display filter is then copied to the clipboard.

Menu Item	Accelerator	Description

Find Packet...	Ctrl+F	This menu item brings up a dialog box that allows you to find a packet by many criteria. There is further information on finding packets in Section 6.8, “Finding packets”.
Find Next	Ctrl+N	This menu item tries to find the next packet matching the settings from "Find Packet...".
Find Previous	Ctrl+B	This menu item tries to find the previous packet matching the settings from "Find Packet...".

Mark Packet (toggle)	Ctrl+M	This menu item "marks" the currently selected packet. See Section 6.10, “Marking packets” for details.
Find Next Mark	Shift+Ctrl+N	Find the next marked packet.
Find Previous Mark	Shift+Ctrl+B	Find the previous marked packet.
Mark All Packets		This menu item "marks" all packets.
Unmark All Packets		This menu item "unmarks" all marked packets.

Set Time Reference (toggle)	Ctrl+T	This menu item set a time reference on the currently selected packet. See Section 6.11.1, “Packet time referencing” for more information about the time referenced packets.
Find Next Reference		This menu item tries to find the next time referenced packet.
Find Previous Reference		This menu item tries to find the previous time referenced packet.

Configuration Profiles...	Shift-Ctrl-A	This menu item brings up a dialog box for handling configuration profiles. More detail is provided in Section 9.6, “Configuration Profiles”.
Preferences...	Shift+Ctrl+P	This menu item brings up a dialog box that allows you to set preferences for many parameters that control Wireshark. You can also save your preferences so Wireshark will use them the next time you start it. More detail is provided in Section 9.5, “Preferences”.

3.7. The "View" menu

The Wireshark View menu contains the fields shown in Table 3.4, “View menu items”.

Figure 3.5. The "View" Menu

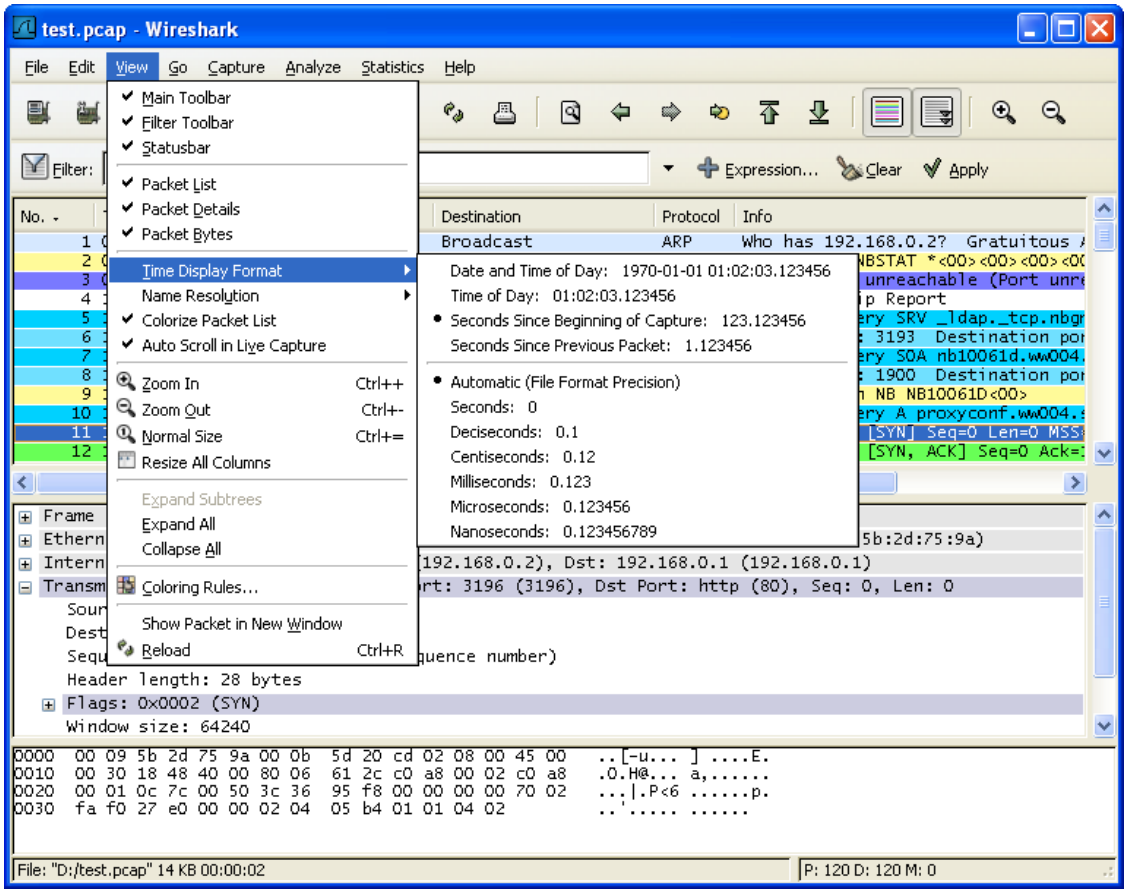


Table 3.4. View menu items

Menu Item	Accelerator	Description
Main Toolbar		This menu item hides or shows the main toolbar, see Section 3.14, “The "Main" toolbar”.
Filter Toolbar		This menu item hides or shows the filter toolbar, see Section 3.15, “The "Filter" toolbar”.
Statusbar		This menu item hides or shows the statusbar, see Section 3.19, “The Statusbar”.

Packet List		This menu item hides or shows the packet list pane, see Sec-

Menu Item	Accelerator	Description
		tion 3.16, “The "Packet List" pane”.
Packet Details		This menu item hides or shows the packet details pane, see Section 3.17, “The "Packet Details" pane”.
Packet Bytes		This menu item hides or shows the packet bytes pane, see Section 3.18, “The "Packet Bytes" pane”.

Time Display Format > Date and Time of Day: 1970-01-01 01:02:03.123456		Selecting this tells Wireshark to display the time stamps in date and time of day format, see Section 6.11, “Time display formats and time references”. **Note!** The fields "Time of Day", "Date and Time of Day", "Seconds Since Beginning of Capture", "Seconds Since Previous Captured Packet" and "Seconds Since Previous Displayed Packet" are mutually exclusive.
Time Display Format > Time of Day: 01:02:03.123456		Selecting this tells Wireshark to display time stamps in time of day format, see Section 6.11, “Time display formats and time references”.
Time Display Format > Seconds Since Beginning of Capture: 123.123456		Selecting this tells Wireshark to display time stamps in seconds since beginning of capture format, see Section 6.11, “Time display formats and time references”.
Time Display Format > Seconds Since Previous Captured Packet: 1.123456		Selecting this tells Wireshark to display time stamps in seconds since previous captured packet format, see Section 6.11, “Time display formats and time references”.
Time Display Format > Seconds Since Previous Displayed Packet: 1.123456		Selecting this tells Wireshark to display time stamps in seconds since previous displayed packet format, see Section 6.11, “Time display formats and time references”.
Time Display Format > Seconds Since Epoch (1970-01-01): 1234567890.123456		Selecting this tells Wireshark to display time stamps in seconds since 1970-01-01 00:00:00, see Section 6.11, “Time display formats and time references”.
Time Display Format > ------		

Menu Item	Accelerator	Description
Time Display Format > Automatic (File Format Precision)		Selecting this tells Wireshark to display time stamps with the precision given by the capture file format used, see Section 6.11, "Time display formats and time references". **Note!** The fields "Automatic", "Seconds" and "...seconds" are mutually exclusive.
Time Display Format > Seconds: 0		Selecting this tells Wireshark to display time stamps with a precision of one second, see Section 6.11, "Time display formats and time references".
Time Display Format > ...seconds: 0....		Selecting this tells Wireshark to display time stamps with a precision of one second, decisecond, centisecond, millisecond, microsecond or nanosecond, see Section 6.11, "Time display formats and time references".
Name Resolution > Resolve Name		This item allows you to trigger a name resolve of the current packet only, see Section 7.7, "Name Resolution".
Name Resolution > Enable for MAC Layer		This item allows you to control whether or not Wireshark translates MAC addresses into names, see Section 7.7, "Name Resolution".
Name Resolution > Enable for Network Layer		This item allows you to control whether or not Wireshark translates network addresses into names, see Section 7.7, "Name Resolution".
Name Resolution > Enable for Transport Layer		This item allows you to control whether or not Wireshark translates transport addresses into names, see Section 7.7, "Name Resolution".
Colorize Packet List		This item allows you to control whether or not Wireshark should colorize the packet list. **Note!** Enabling colorization will slow down the display of new packets while capturing / loading capture files.
Auto Scroll in Live Capture		This item allows you to specify that Wireshark should scroll the packet list pane as new packets come in, so you are always looking at the last packet. If you do not specify this, Wireshark simply adds new packets onto the end of the list, but does not scroll the packet list pane.

Zoom In	Ctrl++	Zoom into the packet data (increase the font size).

Menu Item	Accelerator	Description
Zoom Out	Ctrl+-	Zoom out of the packet data (decrease the font size).
Normal Size	Ctrl+=	Set zoom level back to 100% (set font size back to normal).
Resize All Columns		Resize all column widths so the content will fit into it. **Note!** Resizing may take a significant amount of time, especially if a large capture file is loaded.

Expand Subtrees		This menu item expands the currently selected subtree in the packet details tree.
Expand All		Wireshark keeps a list of all the protocol subtrees that are expanded, and uses it to ensure that the correct subtrees are expanded when you display a packet. This menu item expands all subtrees in all packets in the capture.
Collapse All		This menu item collapses the tree view of all packets in the capture list.

Colorize Conversation		This menu item brings up a submenu that allows you to color packets in the packet list pane based on the addresses of the currently selected packet. This makes it easy to distinguish packets belonging to different conversations. Section 9.3, “Packet colorization”.
Colorize Conversation > Color 1-10		These menu items enable one of the ten temporary color filters based on the currently selected conversation.
Colorize Conversation > Reset coloring		This menu item clears all temporary coloring rules.
Colorize Conversation > New Coloring Rule...		This menu item opens a dialog window in which a new permanent coloring rule can be created based on the currently selected conversation.
Coloring Rules...		This menu item brings up a dialog box that allows you to color packets in the packet list pane according to filter expressions you choose. It can be very useful for spotting certain types of packets, see Section 9.3, “Packet colorization”.

Show Packet in New Window		This menu item brings up the selected packet in a separate window. The separate window shows only the tree view and byte view panes.

Menu Item	Accelerator	Description
Reload	Ctrl-R	This menu item allows you to reload the current capture file.

3.8. The "Go" menu

The Wireshark Go menu contains the fields shown in Table 3.5, “Go menu items”.

Figure 3.6. The "Go" Menu

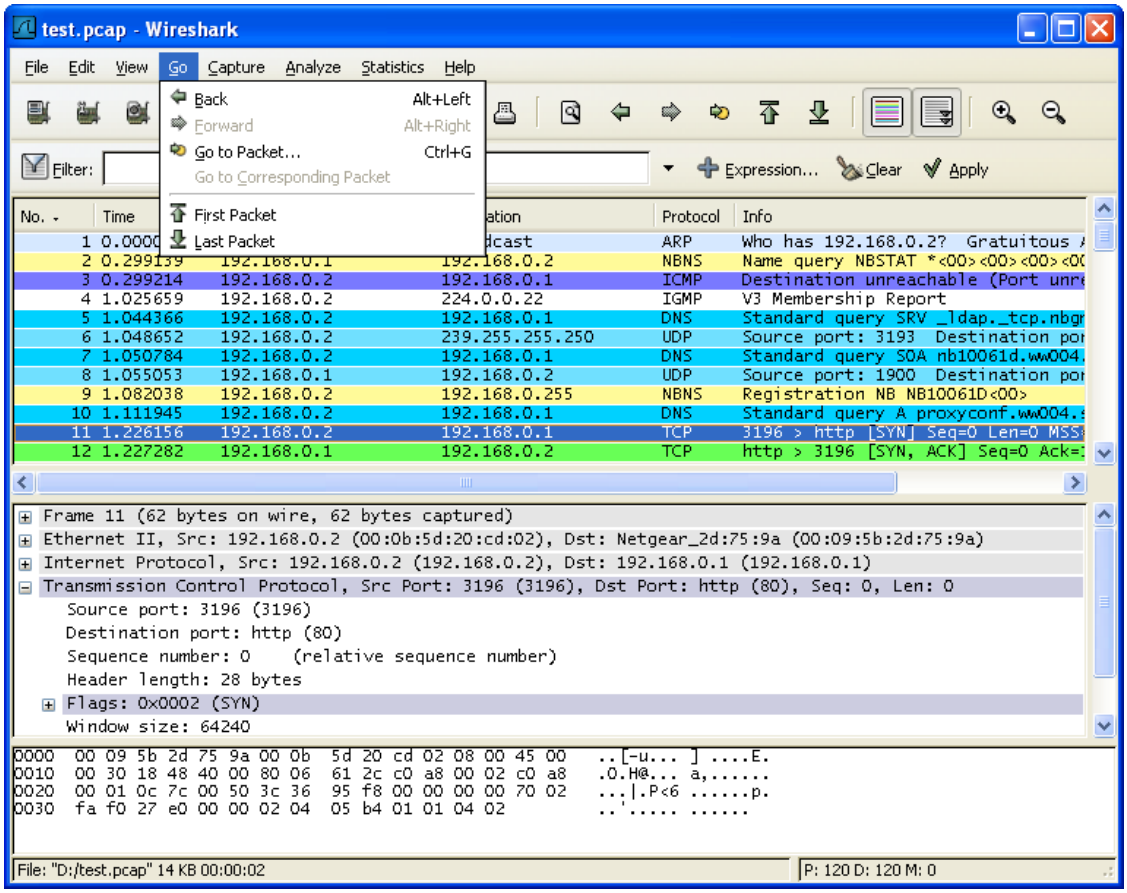

test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Back Alt+Left
Forward Alt+Right
Go to Packet... Ctrl+G
Go to Corresponding Packet
First Packet
Last Packet
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.0000 Broadcast ARP Who has 192.168.0.2? Gratuitous
2 0.299139 192.168.0.1 192.168.0.2 NBNS Name query NBSTAT
3 0.299214 192.168.0.2 192.168.0.1 ICMP Destination unreachable (Port unr
4 1.025659 192.168.0.2 224.0.0.22 IGMP V3 Membership Report
5 1.044366 192.168.0.2 192.168.0.1 DNS Standard query SRV _ldap._tcp.nbg
6 1.048652 192.168.0.2 239.255.255.250 UDP Source port: 3193 Destination por
7 1.050784 192.168.0.2 192.168.0.1 DNS Standard query SOA nb10061d.ww004
8 1.055053 192.168.0.1 192.168.0.2 UDP Source port: 1900 Destination por
9 1.082038 192.168.0.2 192.168.0.255 NBNS Registration NB NB10061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.ww004.
11 1.226156 192.168.0.2 192.168.0.1 TCP 3196 > http [SYN] Seq=0 Len=0 MSS
12 1.227282 192.168.0.1 192.168.0.2 TCP http > 3196 [SYN, ACK] Seq=0 Ack=
Frame 11 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: 3196 (3196)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x0002 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00 ..[-u...] ....E.
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8 .0.H@... a,.....
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02 ...|.P<6 ......p.
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02 ..'..... ......
File: "D:/test.pcap" 14 KB 00:00:02
P: 120 D: 120 M: 0

Table 3.5. Go menu items

Menu Item	Accelerator	Description
Back	Alt+Left	Jump to the recently visited packet in the packet history, much like the page history in a web browser.
Forward	Alt+Right	Jump to the next visited packet in the packet history, much like the page history in a web browser.
Go to Packet...	Ctrl-G	Bring up a dialog box that allows you to specify a packet number, and then goes to that packet. See Section 6.9, “Go to a specific packet” for details.
Go to Corresponding Packet		Go to the corresponding packet of the currently selected protocol

Menu Item	Accelerator	Description
		field. If the selected field doesn't correspond to a packet, this item is greyed out.

Previous Packet	Ctrl+Up	Move to the previous packet in the list. This can be used to move to the previous packet even if the packet list doesn't have keyboard focus.
Next Packet	Ctrl+Down	Move to the next packet in the list. This can be used to move to the previous packet even if the packet list doesn't have keyboard focus.
First Packet		Jump to the first packet of the capture file.
Last Packet		Jump to the last packet of the capture file.

3.9. The "Capture" menu

The Wireshark Capture menu contains the fields shown in Table 3.6, “Capture menu items”.

Figure 3.7. The "Capture" Menu

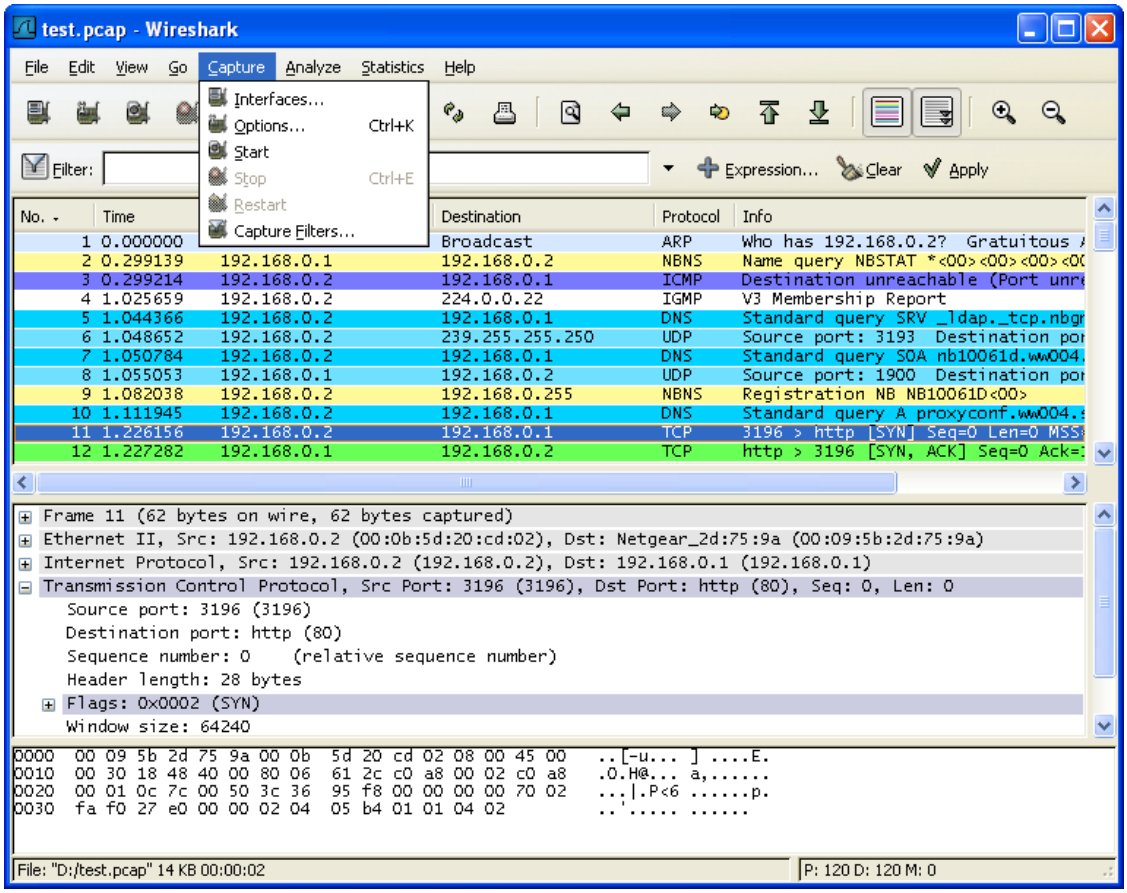

test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Interfaces...
Options... Ctrl+K
Start
Stop Ctrl+E
Restart
Capture Filters...
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 Broadcast ARP Who has 192.168.0.2? Gratuitous
2 0.299139 192.168.0.1 192.168.0.2 NBNS Name query NBSTAT
3 0.299214 192.168.0.2 192.168.0.1 ICMP Destination unreachable (Port
4 1.025659 192.168.0.2 224.0.0.22 IGMP V3 Membership Report
5 1.044366 192.168.0.2 192.168.0.1 DNS Standard query SRV _ldap._tcp.
6 1.048652 192.168.0.2 239.255.255.250 UDP Source port: 3193 Destination
7 1.050784 192.168.0.2 192.168.0.1 DNS Standard query SOA
8 1.055053 192.168.0.1 192.168.0.2 UDP Source port: 1900 Destination
9 1.082038 192.168.0.2 192.168.0.255 NBNS Registration NB NB10061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.
11 1.226156 192.168.0.2 192.168.0.1 TCP 3196 > http [SYN] Seq=0 Len=0
12 1.227282 192.168.0.1 192.168.0.2 TCP http > 3196 [SYN, ACK] Seq=0 Ack=
Frame 11 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: 3196 (3196)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x0002 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02
File: "D:/test.pcap" 14 KB 00:00:02
P: 120 D: 120 M: 0

Table 3.6. Capture menu items

Menu Item	Accelerator	Description
Interfaces...		This menu item brings up a dialog box that shows what's going on at the network interfaces Wireshark knows of, see Section 4.4, “The "Capture Interfaces" dialog box”) .
Options...	Ctrl+K	This menu item brings up the Capture Options dialog box (discussed further in Section 4.5, “The "Capture Options" dialog box”) and allows you to start capturing packets.
Start		Immediately start capturing packets with the same settings than the last time.

Menu Item	Accelerator	Description
Stop	Ctrl+E	This menu item stops the currently running capture, see Section 4.10.1, “Stop the running capture”) .
Restart		This menu item stops the currently running capture and starts again with the same options, this is just for convenience.
Capture Filters...		This menu item brings up a dialog box that allows you to create and edit capture filters. You can name filters, and you can save them for future use. More detail on this subject is provided in Section 6.6, “Defining and saving filters”

3.10. The "Analyze" menu

The Wireshark Analyze menu contains the fields shown in Table 3.7, “Analyze menu items”.

Figure 3.8. The "Analyze" Menu

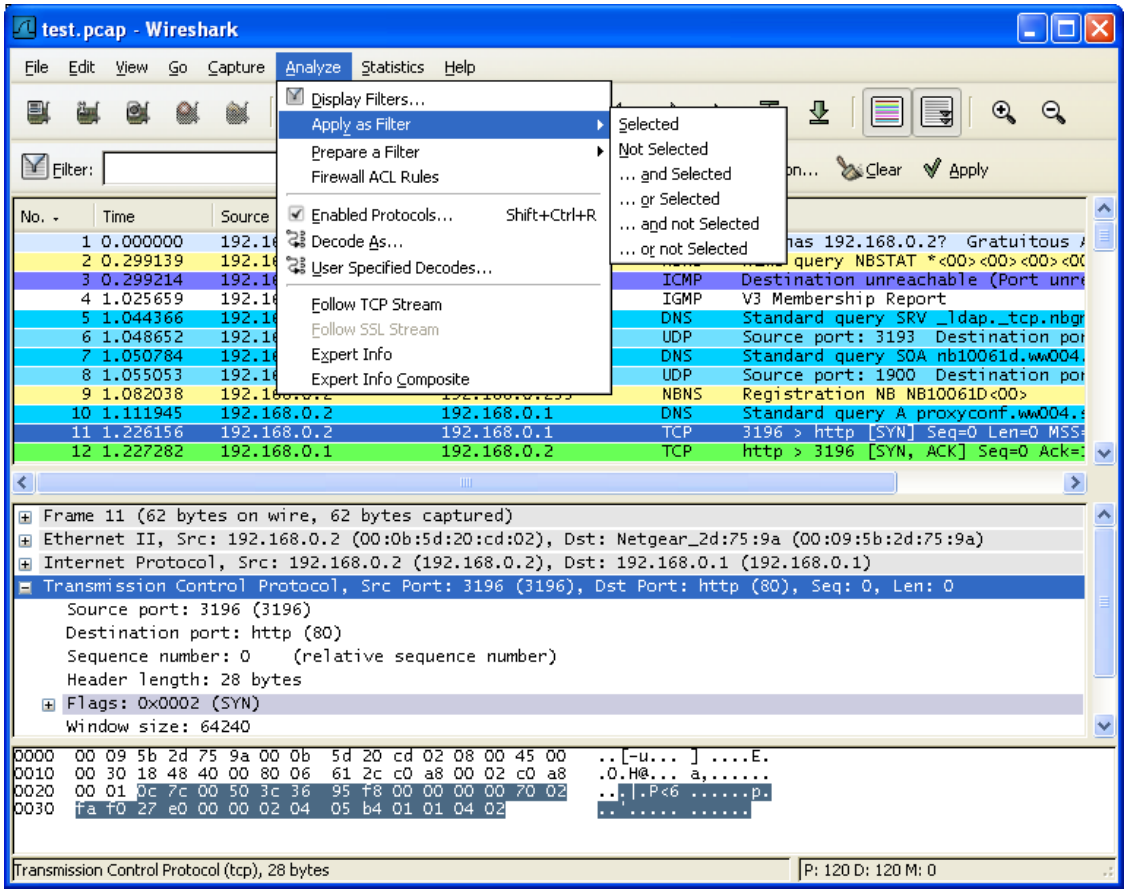

test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Display Filters...
Apply as Filter
Prepare a Filter
Firewall ACL Rules
Enabled Protocols... Shift+Ctrl+R
Decode As...
User Specified Decodes...
Follow TCP Stream
Follow SSL Stream
Expert Info
Expert Info Composite
Selected
Not Selected
... and Selected
... or Selected
... and not Selected
... or not Selected
Filter:
Clear
Apply
No. Time Source
1 0.000000
2 0.299139
3 0.299214
4 1.025659
5 1.044366
6 1.048652
7 1.050784
8 1.055053
9 1.082038
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.ww004.
11 1.226156 192.168.0.2 192.168.0.1 TCP 3196 > http [SYN] Seq=0 Len=0 MSS=
12 1.227282 192.168.0.1 192.168.0.2 TCP http > 3196 [SYN, ACK] Seq=0 Ack=
ICMP Destination unreachable (Port unr
IGMP V3 Membership Report
DNS Standard query SRV _ldap._tcp.nbgr
UDP Source port: 3193 Destination por
DNS Standard query SOA nb10061d.ww004.
UDP Source port: 1900 Destination por
NBNS Registration NB NB10061D<00>
Frame 11 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: 3196 (3196)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x0002 (SYN)
Window size: 64240
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02
Transmission Control Protocol (tcp), 28 bytes
P: 120 D: 120 M: 0

Table 3.7. Analyze menu items

Menu Item	Accelerator	Description
Display Filters...		This menu item brings up a dialog box that allows you to create and edit display filters. You can name filters, and you can save them for future use. More detail on this subject is provided in Section 6.6, “Defining and saving filters”
Display Filter Macros...		This menu item brings up a dialog box that allows you to create and edit display filter macros. You can name filter macros, and you can save them for future use. More detail on this subject is provided in Section 6.7, “Defining and saving filter macros”
Apply as Filter > ...		These menu items will change the current display filter and apply the changed filter immediately. Depending on the chosen menu

Menu Item	Accelerator	Description
		item, the current display filter string will be replaced or appended to by the selected protocol field in the packet details pane.
Prepare a Filter > ...		These menu items will change the current display filter but won't apply the changed filter. Depending on the chosen menu item, the current display filter string will be replaced or appended to by the selected protocol field in the packet details pane.

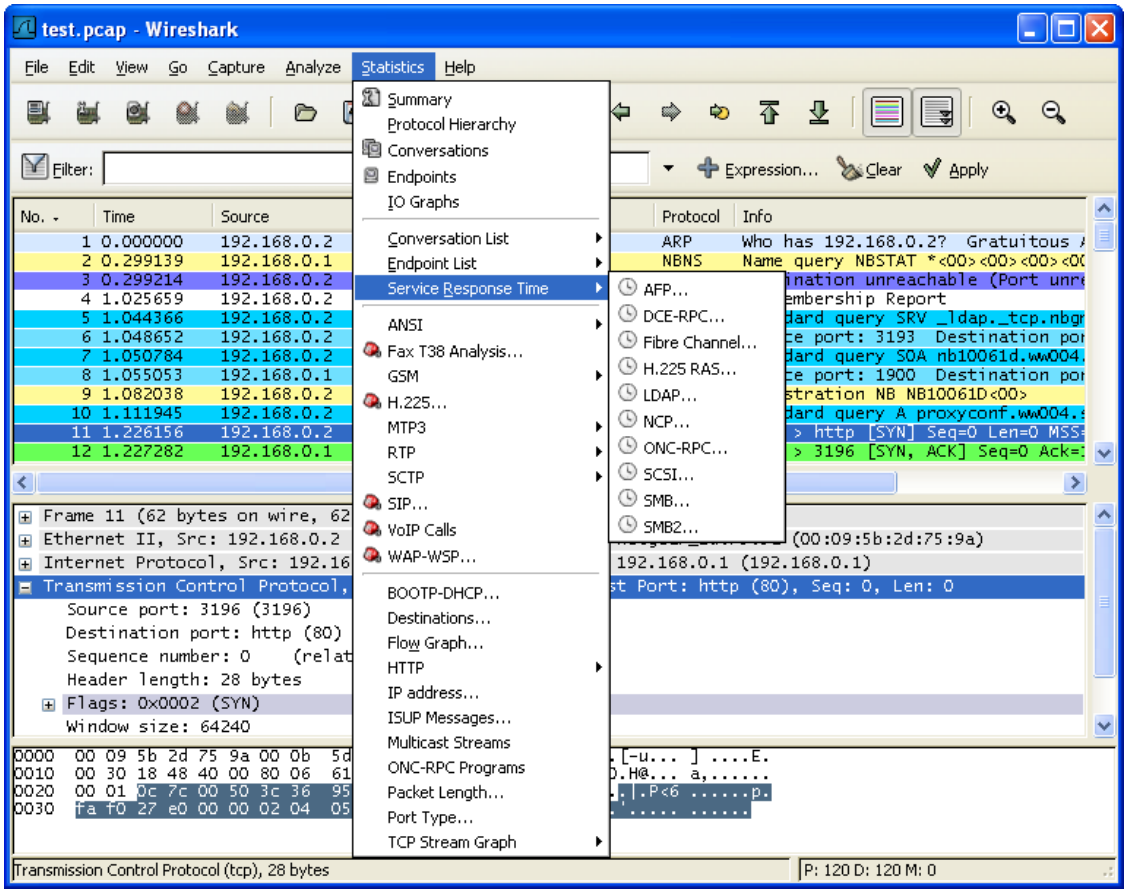
Enabled Protocols...	Shift+Ctrl+R	This menu item allows the user to enable/disable protocol dissectors, see Section 9.4.1, “The "Enabled Protocols" dialog box”
Decode As...		This menu item allows the user to force Wireshark to decode certain packets as a particular protocol, see Section 9.4.2, “User Specified Decodes”
User Specified Decodes...		This menu item allows the user to force Wireshark to decode certain packets as a particular protocol, see Section 9.4.3, “Show User Specified Decodes”

Follow TCP Stream		This menu item brings up a separate window and displays all the TCP segments captured that are on the same TCP connection as a selected packet, see Section 7.2, “Following TCP streams”
Follow UDP Stream		Same functionality as "Follow TCP Stream" but for UDP streams.
Follow SSL Stream		Same functionality as "Follow TCP Stream" but for SSL streams. XXX - how to provide the SSL keys?
Expert Info		Open a dialog showing some expert information about the captured packets in a log style display. The amount of information will depend on the protocol and varies from very detailed to none existing. This is currently a work in progress. XXX - add a new section about this and link from here
Expert Info Composite		Same information as in "Expert Info" but trying to group items together for faster analysis.
Conversation Filter > ...		In this menu you will find conversation filter for various protocols.

3.11. The "Statistics" menu

The Wireshark Statistics menu contains the fields shown in Table 3.8, “Statistics menu items”.

Figure 3.9. The "Statistics" Menu



All menu items will bring up a new window showing specific statistical information.

Table 3.8. Statistics menu items

Menu Item	Accelerator	Description
Summary		Show information about the data captured, see Section 8.2, “The "Summary" window”.
Protocol Hierarchy		Display a hierarchical tree of protocol statistics, see Section 8.3, “The "Protocol Hierarchy" window”.
Conversations		Display a list of conversations (traffic between two endpoints), see Section 8.4.2, “The "Conversations" window”.

Menu Item	Accelerator	Description
Endpoints		Display a list of endpoints (traffic to/from an address), see Section 8.5.2, “The "Endpoints" window”.
IO Graphs		Display user specified graphs (e.g. the number of packets in the course of time), see Section 8.6, “The "IO Graphs" window”.

Conversation List		Display a list of conversations, obsoleted by the combined window of Conversations above, see Section 8.4.3, “The protocol specific "Conversation List" windows”.
Endpoint List		Display a list of endpoints, obsoleted by the combined window of Endpoints above, see Section 8.5.3, “The protocol specific "Endpoint List" windows”.
Service Response Time		Display the time between a request and the corresponding response, see Section 8.10, “Service Response Time”.

ANSI		See Section 8.11, “The protocol specific statistics windows”
Fax T38 Analysis...		See Section 8.11, “The protocol specific statistics windows”
GSM		See Section 8.11, “The protocol specific statistics windows”
H.225...		See Section 8.11, “The protocol specific statistics windows”
LTE MAC		See Section 8.9, “LTE MAC Traffic Statistics”
MTP3		See Section 8.11, “The protocol specific statistics windows”
RTP		See Section 8.11, “The protocol specific statistics windows”
SCTP		See Section 8.11, “The protocol specific statistics windows”
SIP		See Section 8.11, “The protocol specific statistics windows”
VoIP Calls...		See Section 8.11, “The protocol specific statistics windows”
WAP-WSP...		See Section 8.11, “The protocol specific statistics windows”

BOOTP-DHCP...		See Section 8.11, “The protocol specific statistics windows”
Flow Graph...		See Section 8.11, “The protocol specific statistics windows”
HTTP		HTTP request/response statistics, see Section 8.11, “The protocol specific statistics windows”

Menu Item	Accelerator	Description
IP Addresses...		See Section 8.11, “The protocol specific statistics windows”
IP Destinations...		See Section 8.11, “The protocol specific statistics windows”
IP Protocol Types...		See Section 8.11, “The protocol specific statistics windows”
ISUP Messages		See Section 8.11, “The protocol specific statistics windows”
ONC-RPC Programs		See Section 8.11, “The protocol specific statistics windows”
Packet Lengths...		See Section 8.11, “The protocol specific statistics windows”
SMPP Operations...		See Section 8.11, “The protocol specific statistics windows”
TCP Stream Graph		See Section 8.11, “The protocol specific statistics windows”
UCP Messages...		See Section 8.11, “The protocol specific statistics windows”
UDP Multicast Streams		See Section 8.11, “The protocol specific statistics windows”
WLAN Traffic		See Section 8.8, “WLAN Traffic Statistics”

3.12. The "Tools" menu

The Wireshark Tools menu contains the fields shown in Table 3.9, “Tools menu items”.

Table 3.9. Tools menu items

Menu Item	Accelerator	Description
Firewall ACL Rules		This allows you to create command-line ACL rules for many different firewall products, including Cisco IOS, Linux Netfilter (iptables), OpenBSD pf and Windows Firewall (via netsh). Rules for MAC addresses, IPv4 addresses, TCP and UDP ports, and IPv4+port combinations are supported. It is assumed that the rules will be applied to an outside interface.

3.13. The "Help" menu

The Wireshark Help menu contains the fields shown in Table 3.10, “Help menu items”.

Figure 3.10. The "Help" Menu

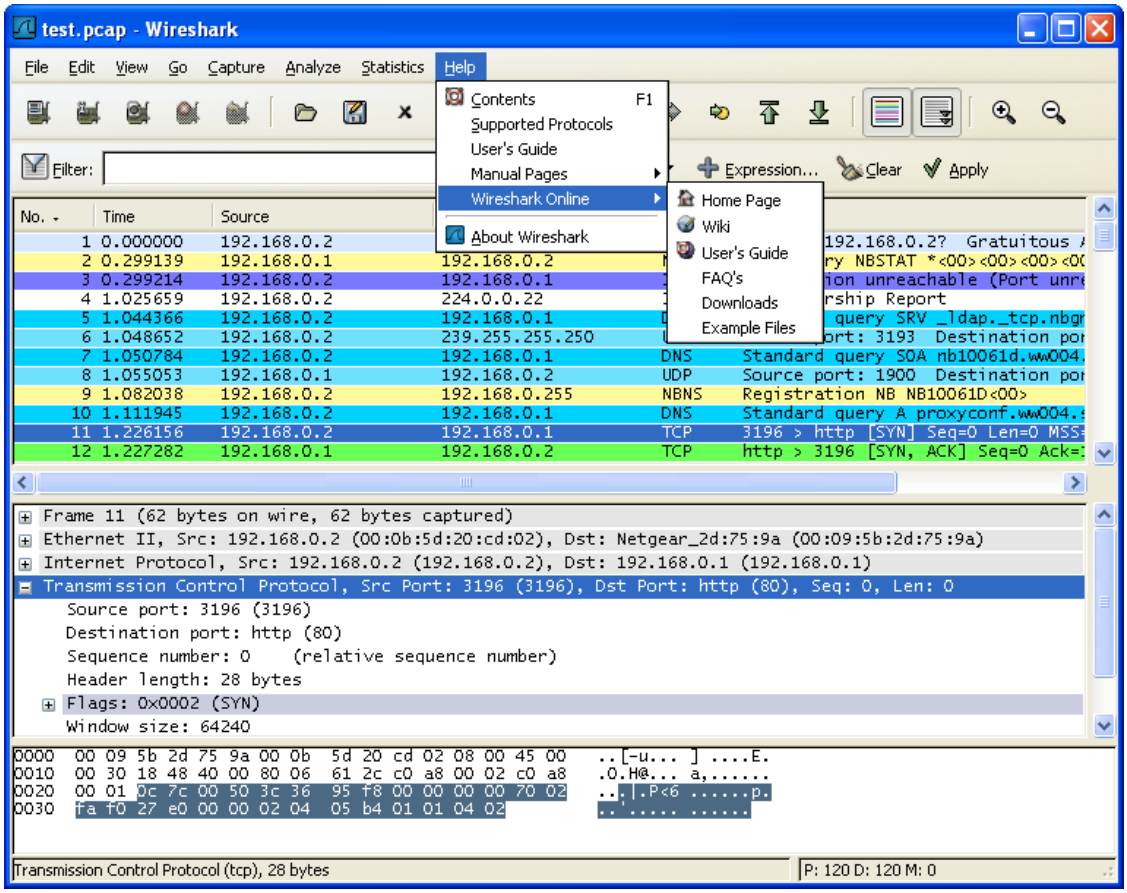


Table 3.10. Help menu items

Menu Item	Accelerator	Description
Contents	F1	This menu item brings up a basic help system.
FAQ's		This menu item starts a Web browser showing various FAQ's.
Manual Pages > ...		This menu item starts a Web browser showing one of the locally installed html manual pages.

Wireshark Online > ...		This menu item starts a Web browser showing the chosen webpage from: http://www.wireshark.org.

Menu Item	Accelerator	Description
Supported Protocols (slow!)		This menu item brings up a dialog box showing the supported protocols and protocol fields.

About Wireshark		This menu item brings up an information window that provides some information on Wireshark, such as the plugins, the used folders, ...

Note!

Calling a Web browser might be unsupported in your version of Wireshark. If this is the case, the corresponding menu items will be hidden.

Note!

If calling a Web browser fails on your machine, maybe because just nothing happens or the browser is started but no page is shown, have a look at the web browser setting in the preferences dialog.

3.14. The "Main" toolbar

The main toolbar provides quick access to frequently used items from the menu. This toolbar cannot be customized by the user, but it can be hidden using the View menu, if the space on the screen is needed to show even more packet data.

As in the menu, only the items useful in the current program state will be available. The others will be greyed out (e.g. you cannot save a capture file if you haven't loaded one).

Figure 3.11. The "Main" toolbar

Table 3.11. Main toolbar items

Toolbar Icon	Toolbar Item	Corresponding Menu Item	Description
	Interfaces...	Capture/Interfaces...	This item brings up the Capture Interfaces List dialog box (discussed further in Section 4.3, “Start Capturing”).
	Options...	Capture/Options...	This item brings up the Capture Options dialog box (discussed further in Section 4.3, “Start Capturing”) and allows you to start capturing packets.
	Start	Capture/Start	This item starts capturing packets with the options form the last time.
	Stop	Capture/Stop	This item stops the currently running live capture process Section 4.3, “Start Capturing”).
	Restart	Capture/Restart	This item stops the currently running live capture process and restarts it again, for convenience.

	Open...	File/Open...	This item brings up the file open dialog box that allows you to load a capture file for viewing. It is discussed in more detail in Section 5.2.1, “The "Open Capture File" dialog box”.
	Save As...	File/Save As...	This item allows you to save the current capture file to whatever file you would like. It pops up the Save Capture File As dialog box (which is discussed further in Section 5.3.1, “The "Save Capture File As" dialog box”). **Note!** If you currently have a temporary cap-

Toolbar Icon	Toolbar Item	Corresponding Menu Item	Description
			ture file, the Save icon will be shown instead.
	Close	File/Close	This item closes the current capture. If you have not saved the capture, you will be asked to save it first.
	Reload	View/Reload	This item allows you to reload the current capture file.
	Print...	File/Print...	This item allows you to print all (or some of) the packets in the capture file. It pops up the Wireshark Print dialog box (which is discussed further in Section 5.7, "Printing packets").

	Find Packet...	Edit/Find Packet...	This item brings up a dialog box that allows you to find a packet. There is further information on finding packets in Section 6.8, "Finding packets".
	Go Back	Go/Go Back	This item jumps back in the packet history.
	Go Forward	Go/Go Forward	This item jumps forward in the packet history.
	Go to Packet...	Go/Go to Packet...	This item brings up a dialog box that allows you to specify a packet number to go to that packet.
	Go To First Packet	Go/First Packet	This item jumps to the first packet of the capture file.
	Go To Last Packet	Go/Last Packet	This item jumps to the last packet of the capture file.

	Colorize	View/Colorize	Colorize the packet list (or not).
	Auto Scroll in Live Capture	View/Auto Scroll in Live Capture	Auto scroll packet list while doing a live capture (or not).

	Zoom In	View/Zoom In	Zoom into the packet data (increase the font size).
	Zoom Out	View/Zoom Out	Zoom out of the packet data (decrease the font size).
	Normal Size	View/Normal Size	Set zoom level back to 100%.

Toolbar Icon	Toolbar Item	Corresponding Menu Item	Description
	Resize Columns	View/Resize Columns	Resize columns, so the content fits into them.

	Capture Filters...	Capture/Capture Filters...	This item brings up a dialog box that allows you to create and edit capture filters. You can name filters, and you can save them for future use. More detail on this subject is provided in Section 6.6, “Defining and saving filters”.
	Display Filters...	Analyze/Display Filters...	This item brings up a dialog box that allows you to create and edit display filters. You can name filters, and you can save them for future use. More detail on this subject is provided in Section 6.6, “Defining and saving filters”.
	Coloring Rules...	View/Coloring Rules...	This item brings up a dialog box that allows you color packets in the packet list pane according to filter expressions you choose. It can be very useful for spotting certain types of packets. More detail on this subject is provided in Section 9.3, “Packet colorization”.
	Preferences...	Edit/Preferences	This item brings up a dialog box that allows you to set preferences for many parameters that control Wireshark. You can also save your preferences so Wireshark will use them the next time you start it. More detail is provided in Section 9.5, “Preferences”

	Help	Help/Contents	This item brings up help dialog box.

3.15. The "Filter" toolbar

The filter toolbar lets you quickly edit and apply display filters. More information on display filters is available in Section 6.3, “Filtering packets while viewing”.

Figure 3.12. The "Filter" toolbar

Table 3.12. Filter toolbar items

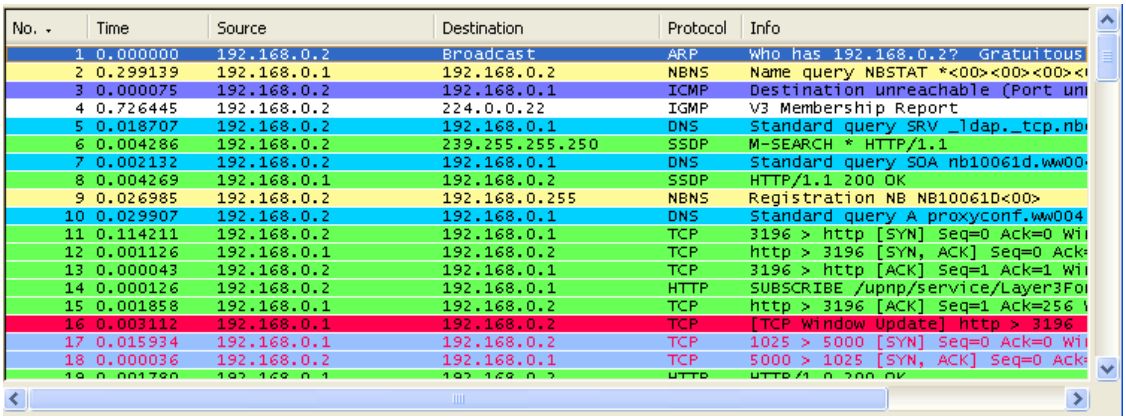
Toolbar Icon	Toolbar Item	Description
	Filter:	Brings up the filter construction dialog, described in Figure 6.7, “The "Capture Filters" and "Display Filters" dialog boxes”.
	Filter input	The area to enter or edit a display filter string, see Section 6.4, “Building display filter expressions” . A syntax check of your filter string is done while you are typing. The background will turn red if you enter an incomplete or invalid string, and will become green when you enter a valid string. You can click on the pull down arrow to select a previously-entered filter string from a list. The entries in the pull down list will remain available even after a program restart. **Note!** After you've changed something in this field, don't forget to press the Apply button (or the Enter/Return key), to apply this filter string to the display. **Note!** This field is also where the current filter in effect is displayed.
	Expression...	The middle button labeled "Add Expression..." opens a dialog box that lets you edit a display filter from a list of protocol fields, described in Section 6.5, “The "Filter Expression" dialog box”
	Clear	Reset the current display filter and clears the edit area.
	Apply	Apply the current value in the edit area as the new display filter. **Note!** Applying a display filter on large capture files might take quite a long time!

Toolbar Icon	Toolbar Item	Description

3.16. The "Packet List" pane

The packet list pane displays all the packets in the current capture file.

Figure 3.13. The "Packet List" pane



Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the "Packet Details" and "Packet Bytes" panes.

While dissecting a packet, Wireshark will place information from the protocol dissectors into the columns. As higher level protocols might overwrite information from lower levels, you will typically see the information from the highest possible level only.

For example, let's look at a packet containing TCP inside IP inside an Ethernet packet. The Ethernet dissector will write its data (such as the Ethernet addresses), the IP dissector will overwrite this by its own (such as the IP addresses), the TCP dissector will overwrite the IP information, and so on.

There are a lot of different columns available. Which columns are displayed can be selected by preference settings, see Section 9.5, “Preferences”.

The default columns will show:

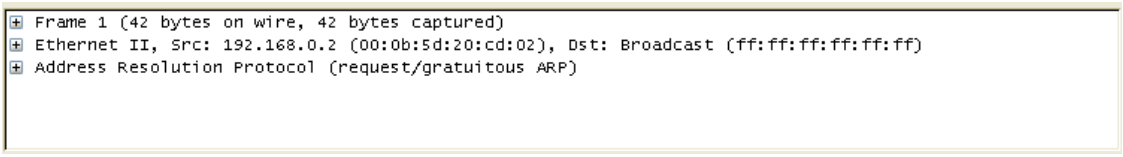
- **No.** The number of the packet in the capture file. This number won't change, even if a display filter is used.
- **Time** The timestamp of the packet. The presentation format of this timestamp can be changed, see Section 6.11, “Time display formats and time references”.
- **Source** The address where this packet is coming from.
- **Destination** The address where this packet is going to.
- **Protocol** The protocol name in a short (perhaps abbreviated) version.
- **Info** Additional information about the packet content.

There is a context menu (right mouse click) available, see details in Figure 6.3, “Pop-up menu of the "Packet List" pane”.

3.17. The "Packet Details" pane

The packet details pane shows the current packet (selected in the "Packet List" pane) in a more detailed form.

Figure 3.14. The "Packet Details" pane


Frame 1 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request/gratuitous ARP)

This pane shows the protocols and protocol fields of the packet selected in the "Packet List" pane. The protocols and fields of the packet are displayed using a tree, which can be expanded and collapsed.

There is a context menu (right mouse click) available, see details in Figure 6.4, “Pop-up menu of the "Packet Details" pane”.

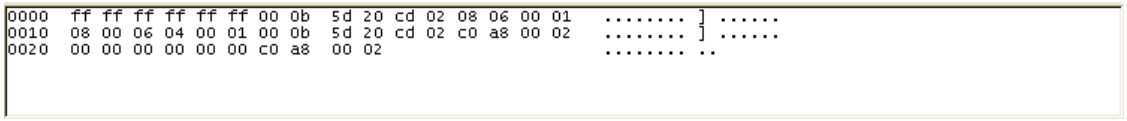
Some protocol fields are specially displayed.

- **Generated fields** Wireshark itself will generate additional protocol fields which are surrounded by brackets. The information in these fields is derived from the known context to other packets in the capture file. For example, Wireshark is doing a sequence/acknowledge analysis of each TCP stream, which is displayed in the [SEQ/ACK analysis] fields of the TCP protocol.
- **Links** If Wireshark detected a relationship to another packet in the capture file, it will generate a link to that packet. Links are underlined and displayed in blue. If double-clicked, Wireshark jumps to the corresponding packet.

3.18. The "Packet Bytes" pane

The packet bytes pane shows the data of the current packet (selected in the "Packet List" pane) in a hex-dump style.

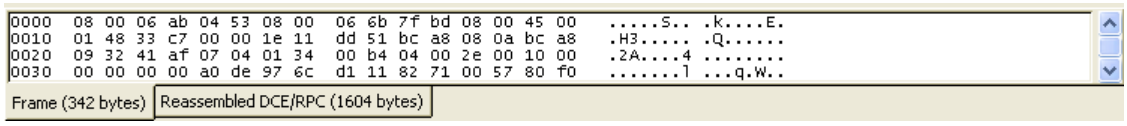
Figure 3.15. The "Packet Bytes" pane



As usual for a hexdump, the left side shows the offset in the packet data, in the middle the packet data is shown in a hexadecimal representation and on the right the corresponding ASCII characters (or . if not appropriate) are displayed.

Depending on the packet data, sometimes more than one page is available, e.g. when Wireshark has re-assembled some packets into a single chunk of data, see Section 7.6, “Packet Reassembling”. In this case there are some additional tabs shown at the bottom of the pane to let you select the page you want to see.

Figure 3.16. The "Packet Bytes" pane with tabs



Note!

The additional pages might contain data picked from multiple packets.

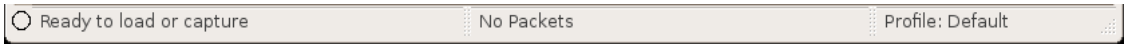
The context menu (right mouse click) of the tab labels will show a list of all available pages. This can be helpful if the size in the pane is too small for all the tab labels.

3.19. The Statusbar

The statusbar displays informational messages.

In general, the left side will show context related information, the middle part will show the current number of packets, and the right side will show the selected configuration profile. Drag the handles between the text areas to change the size.

Figure 3.17. The initial Statusbar



This statusbar is shown while no capture file is loaded, e.g. when Wireshark is started.

Figure 3.18. The Statusbar with a loaded capture file

- **The colorized bullet** on the left shows the highest expert info level found in the currently loaded capture file. Hovering the mouse over this icon will show a textual description of the expert info level, and clicking the icon will bring up the Expert Infos dialog box. For a detailed description of expert info, see Section 7.3, “Expert Infos”.
- **The left side** shows information about the capture file, its name, its size and the elapsed time while it was being captured.
- **The middle part** shows the current number of packets in the capture file. The following values are displayed:
  - *Packets:* the number of captured packets
  - *Displayed:* the number of packets currently being displayed
  - *Marked:* the number of marked packets
  - *Dropped:* the number of dropped packets (only displayed if Wireshark was unable to capture all packets)
- **The right side** shows the selected configuration profile. Clicking in this part of the statusbar will bring up a menu with all available configuration profiles, and selecting from this list will change the configuration profile.

Figure 3.19. The Statusbar with a configuration profile menu

For a detailed description of configuration profiles, see Section 9.6, “Configuration Profiles”.

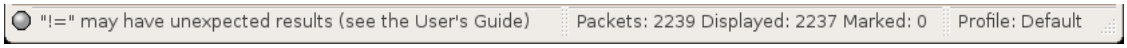
Figure 3.20. The Statusbar with a selected protocol field

This is displayed if you have selected a protocol field from the "Packet Details" pane.

Tip!

The value between the brackets (in this example **arp.opcode**) can be used as a display filter string, representing the selected protocol field.

Figure 3.21. The Statusbar with a display filter message



This is displayed if you are trying to use a display filter which may have unexpected results. For a detailed description, see Section 6.4.4, “A common mistake”.

Chapter 4. Capturing Live Network Data

4.1. Introduction

Capturing live network data is one of the major features of Wireshark.

The Wireshark capture engine provides the following features:

- Capture from different kinds of network hardware (Ethernet, Token Ring, ATM, ...).
- Stop the capture on different triggers like: amount of captured data, captured time, captured number of packets.
- Simultaneously show decoded packets while Wireshark keeps on capturing.
- Filter packets, reducing the amount of data to be captured, see Section 4.9, “Filtering while capturing”.
- Capturing into multiple files while doing a long term capture, and in addition the option to form a ringbuffer of these files, keeping only the last x files, useful for a "very long term" capture, see Section 4.7, “Capture files and file modes”.

The capture engine still lacks the following features:

- Simultaneous capturing from multiple network interfaces (however, you can start multiple instances of Wireshark and merge capture files later).
- Stop capturing (or doing some other action), depending on the captured data.

4.2. Prerequisites

Setting up Wireshark to capture packets for the first time can be tricky.

Tip!

A comprehensive guide "How To setup a Capture" is available at: http://wiki.wireshark.org/CaptureSetup.

Here are some common pitfalls:

- You need to have root / Administrator privileges to start a live capture.
- You need to choose the right network interface to capture packet data from.
- You need to capture at the right place in the network to see the traffic you want to see.
- ... and a lot more!.

If you have any problems setting up your capture environment, you should have a look at the guide mentioned above.

4.3. Start Capturing

One of the following methods can be used to start capturing packets with Wireshark:

- You can get an overview of the available local interfaces using the " " Capture Interfaces" dialog box, see Figure 4.1, “The "Capture Interfaces" dialog box on Microsoft Windows” or Figure 4.2, “The "Capture Interfaces" dialog box on Unix/Linux”. You can start a capture from this dialog box, using (one of) the "Capture" button(s).
- You can start capturing using the " " Capture Options" dialog box, see Figure 4.3, “The "Capture Options" dialog box”.
- If you have selected the right capture options before, you can immediately start a capture using the " " Capture Start" menu / toolbar item. The capture process will start immediately.
- If you already know the name of the capture interface, you can start Wireshark from the command line and use the following:

  ```
  wireshark -i eth0 -k
  ```

  This will start Wireshark capturing on interface eth0, more details can be found at: Section 9.2, “Start Wireshark from the command line”.

4.4. The "Capture Interfaces" dialog box

When you select "Interfaces..." from the Capture menu, Wireshark pops up the "Capture Interfaces" dialog box as shown in Figure 4.1, “The "Capture Interfaces" dialog box on Microsoft Windows” or Figure 4.2, “The "Capture Interfaces" dialog box on Unix/Linux”.

This dialog consumes lot's of system resources!

As the "Capture Interfaces" dialog is showing live captured data, it is consuming a lot of system resources. Close this dialog as soon as possible to prevent excessive system load.

Not all available interfaces may be displayed!

This dialog box will only show the local interfaces Wireshark knows of. It will not show interfaces marked as hidden in the "Interface Options" preferences dialog. As Wireshark might not be able to detect all local interfaces, and it cannot detect the remote interfaces available, there could be more capture interfaces available than listed.

Figure 4.1. The "Capture Interfaces" dialog box on Microsoft Windows

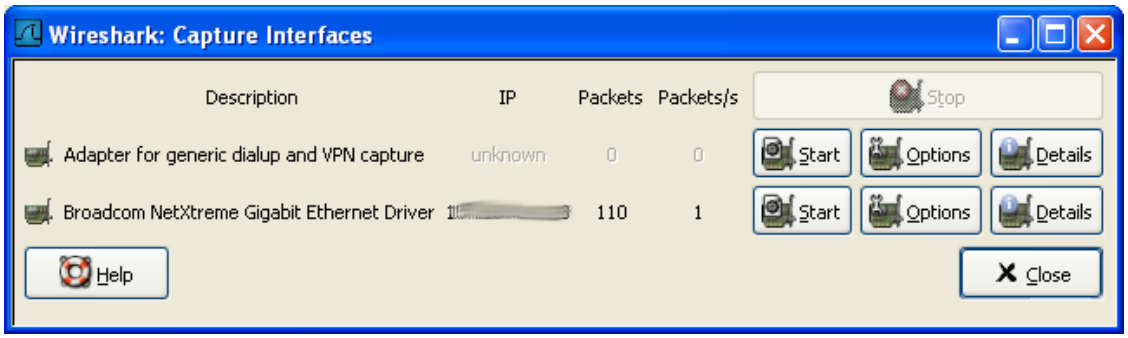
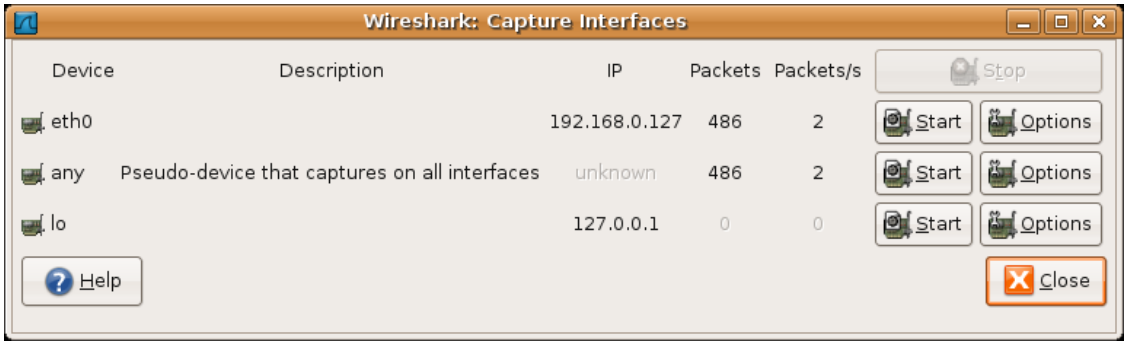


Figure 4.2. The "Capture Interfaces" dialog box on Unix/Linux



Device (Unix/Linux only) The interface device name.

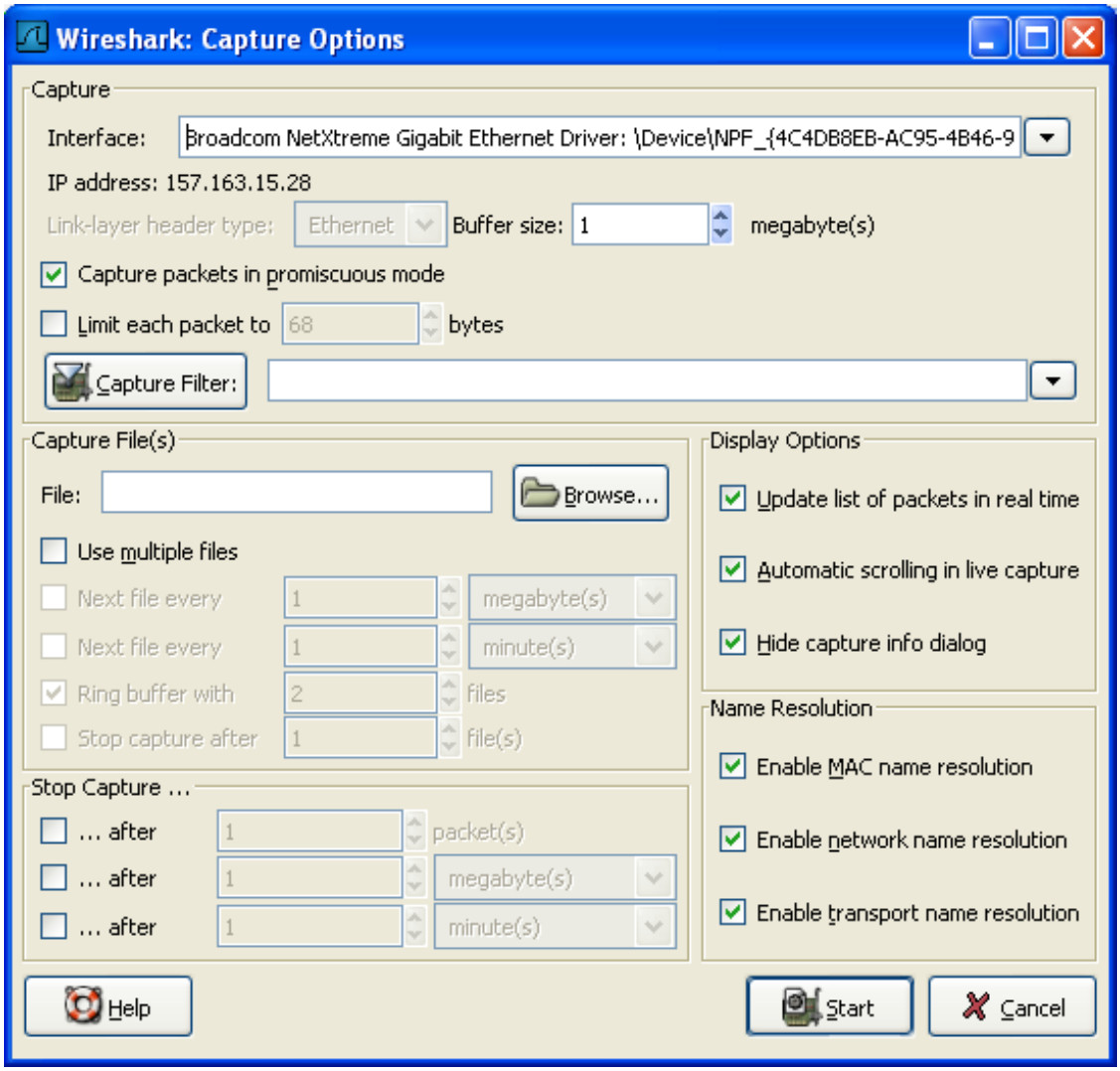
Description The interface description provided by the operating system.

IP	The first IP address Wireshark could resolve from this interface. If no address could be resolved (e.g. no DHCP server available), "unknown" will be displayed. If more than one IP address could be resolved, only the first is shown (unpredictable which one in that case).
Packets	The number of packets captured from this interface, since this dialog was opened. Will be greyed out, if no packet was captured in the last second.
Packets/s	Number of packets captured in the last second. Will be greyed out, if no packet was captured in the last second.
Stop	Stop a currently running capture.
Start	Start a capture on this interface immediately, using the settings from the last capture.
Options	Open the Capture Options dialog with this interface selected, see Section 4.5, “The "Capture Options" dialog box”.
Details (Microsoft Windows only)	Open a dialog with detailed information about the interface, see Section 4.6, “The "Interface Details" dialog box”.
Help	Show this help page.
Close	Close this dialog box.

4.5. The "Capture Options" dialog box

When you select Start... from the Capture menu (or use the corresponding item in the "Main" toolbar), Wireshark pops up the "Capture Options" dialog box as shown in Figure 4.3, “The "Capture Options" dialog box”.

Figure 4.3. The "Capture Options" dialog box



Tip!

If you are unsure which options to choose in this dialog box, just try keeping the defaults as this should work well in many cases.

You can set the following fields in this dialog box:

4.5.1. Capture frame

Interface

This field specifies the interface you want to capture on. You can only capture on one interface, and you can only capture on interfaces that Wireshark has found on the system. It is a drop-down list, so simply click on the button on the right hand side and select the interface you want. It defaults to the first non-loopback interface that supports capturing, and if there are none, the first loopback interface. On some systems, loopback interfaces cannot be used for capturing (loopback interfaces are not available on Windows platforms).

This field performs the same function as the **-i <interface>** command line option.

IP address

The IP address(es) of the selected interface. If no address could be resolved from the system, "unknown" will be shown.

Link-layer header type

Unless you are in the rare situation that you need this, just keep the default. For a detailed description, see Section 4.8, "Link-layer header type"

Buffer size: n megabyte(s)

Enter the buffer size to be used while capturing. This is the size of the kernel buffer which will keep the captured packets, until they are written to disk. If you encounter packet drops, try increasing this value.

Note

This option is only available on Windows platforms.

Capture packets in promiscuous mode

This checkbox allows you to specify that Wireshark should put the interface in promiscuous mode when capturing. If you do not specify this, Wireshark will only capture the packets going to or from your computer (not all packets on your LAN segment).

Note

If some other process has put the interface in promiscuous mode you may be capturing in promiscuous mode even if you turn off this option

Note

Even in promiscuous mode you still won't necessarily see all packets on your LAN segment, see http://www.wireshark.org/faq.html#promiscsniff for some more explanations.

Limit each packet to n bytes

This field allows you to specify the maximum amount of data that will be captured for each packet, and is sometimes referred to as the **snaplen**. If disabled, the default is 65535, which will be sufficient for most protocols. Some rules of thumb:

- If you are unsure, just keep the default value.
- If you don't need all of the data in a packet - for example, if you only need the link-layer, IP, and TCP headers - you might want to choose a small snapshot length, as less CPU time is required for copying packets, less buffer space is required for packets, and thus perhaps fewer packets will be dropped if traffic is very heavy.
- If you don't capture all of the data in a packet, you might find that the packet data you want is in the part that's dropped, or that reassembly isn't possible as the data required for reassembly is missing.

Capture Filter

This field allows you to specify a capture filter. Capture filters are discussed in more details in Section 4.9, “Filtering while capturing”. It defaults to empty, or no filter.

You can also click on the button labeled "Capture Filter", and Wireshark will bring up the Capture Filters dialog box and allow you to create and/or select a filter. Please see Section 6.6, “Defining and saving filters”

4.5.2. Capture File(s) frame

An explanation about capture file usage can be found in Section 4.7, “Capture files and file modes”.

File

This field allows you to specify the file name that will be used for the capture file. This field is left blank by default. If the field is left blank, the capture data will be stored in a temporary file, see Section 4.7, “Capture files and file modes” for details.

You can also click on the button to the right of this field to browse through the filesystem.

Use multiple files

Instead of using a single file, Wireshark will automatically switch to a new one, if a specific trigger condition is reached.

Next file every n megabyte(s)

Multiple files only: Switch to the next file after the given number of byte(s)/kilobyte(s)/megabyte(s)/gigabyte(s) have been captured.

Next file every n minute(s)

Multiple files only: Switch to the next file after the given number of second(s)/minutes(s)/hours(s)/days(s) have elapsed.

Ring buffer with n files

Multiple files only: Form a ring buffer of the capture files, with the given number of files.

Stop capture after n file(s)

Multiple files only: Stop capturing after switching to the next file the given number of times.

4.5.3. Stop Capture... frame

... after n packet(s) Stop capturing after the given number of packets have been captured.

... after n megabytes(s) Stop capturing after the given number of byte(s)/kilobyte(s)/megabyte(s)/gigabyte(s) have been captured. This option is greyed out, if "Use multiple files" is selected.

... after n minute(s) Stop capturing after the given number of second(s)/minutes(s)/hours(s)/days(s) have elapsed.

4.5.4. Display Options frame

Update list of packets in real time This option allows you to specify that Wireshark should update the packet list pane in real time. If you do not specify this, Wireshark does not display any packets until you stop the capture. When you check this, Wireshark captures in a separate process and feeds the captures to the display process.

Automatic scrolling in live capture This option allows you to specify that Wireshark should scroll the packet list pane as new packets come in, so you are always looking at the last packet. If you do not specify this, Wireshark simply adds new packets onto the end of the list, but does not scroll the packet list pane. This option is greyed out if "Update list of packets in real time" is disabled.

Hide capture info dialog If this option is checked, the capture info dialog described in Section 4.10, “While a Capture is running ...” will be hidden.

4.5.5. Name Resolution frame

Enable MAC name resolution This option allows you to control whether or not Wireshark translates MAC addresses into names, see Section 7.7, “Name Resolution”.

Enable network name resolution This option allows you to control whether or not Wireshark translates network addresses into names, see Section 7.7, “Name Resolution”.

Enable transport name resolution This option allows you to control whether or not Wireshark translates transport addresses into protocols, see Section 7.7, “Name Resolution”.

4.5.6. Buttons

Once you have set the values you desire and have selected the options you need, simply click on **Start** to commence the capture, or **Cancel** to cancel the capture.

If you start a capture, Wireshark allows you to stop capturing when you have enough packets captured, for details see Section 4.10, “While a Capture is running ...”.

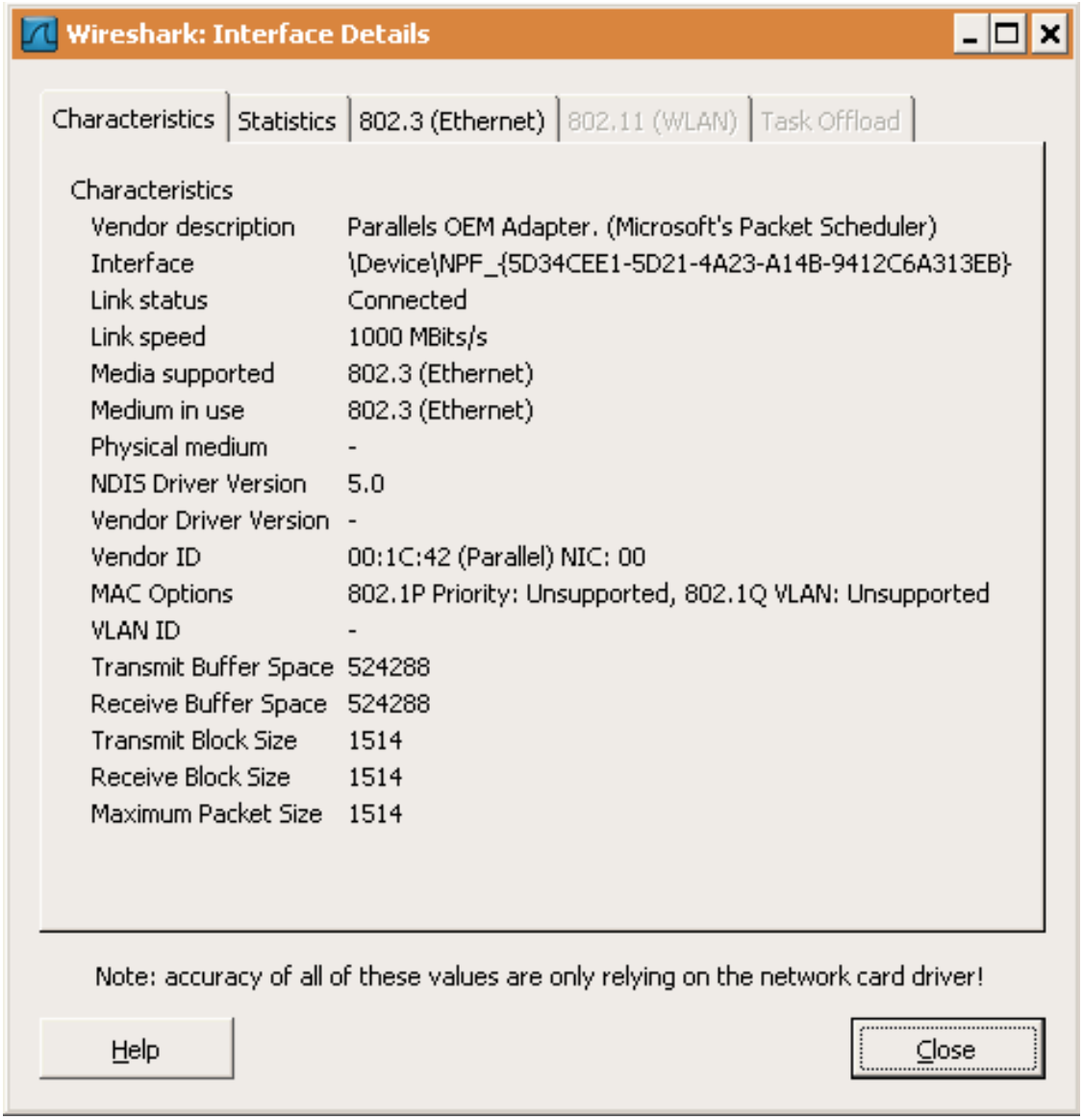
4.6. The "Interface Details" dialog box

When you select Details from the Capture Interface menu, Wireshark pops up the "Interface Details" dialog box as shown in Figure 4.4, “The "Interface Details" dialog box”. This dialog shows various characteristics and statistics for the selected interface.

Microsoft Windows only

This dialog is only available on Microsoft Windows

Figure 4.4. The "Interface Details" dialog box



4.7. Capture files and file modes

While capturing, the underlying libpcap capturing engine will grab the packets from the network card and keep the packet data in a (relatively) small kernel buffer. This data is read by Wireshark and saved into the capture file(s) the user specified.

Different modes of operation are available when saving this packet data to the capture file(s).

Tip!

Working with large files (several 100 MB's) can be quite slow. If you plan to do a long term capture or capturing from a high traffic network, think about using one of the "Multiple files" options. This will spread the captured packets over several smaller files which can be much more pleasant to work with.

Note!

Using Multiple files may cut context related information. Wireshark keeps context information of the loaded packet data, so it can report context related problems (like a stream error) and keeps information about context related protocols (e.g. where data is exchanged at the establishing phase and only referred to in later packets). As it keeps this information only for the loaded file, using one of the multiple file modes may cut these contexts. If the establishing phase is saved in one file and the things you would like to see is in another, you might not see some of the valuable context related information.

Tip!

Information about the folders used for the capture file(s), can be found in Appendix A, *Files and Folders*.

Table 4.1. Capture file mode selected by capture options

"File" option	"Use multiple files" option	"Ring buffer with n files" option	Mode	Resulting filename(s) used
-	-	-	**Single temporary file**	etherXXXXXX (where XXXXXX is a unique number)
foo.cap	-	-	**Single named file**	foo.cap
foo.cap	x	-	**Multiple files, continuous**	foo_00001_20040205110102.cap, foo_00002_20040205110102.cap, ...
foo.cap	x	x	**Multiple files, ring buffer**	foo_00001_20040205110102.cap, foo_00002_20040205110102.cap, ...

Single temporary file A temporary file will be created and used (this is the default).

After the capturing is stopped, this file can be saved later under a user specified name.

Single named file

A single capture file will be used. If you want to place the new capture file to a specific folder, choose this mode.

Multiple files, continuous

Like the "Single named file" mode, but a new file is created and used, after reaching one of the multiple file switch conditions (one of the "Next file every ..." values).

Multiple files, ring buffer

Much like "Multiple files continuous", reaching one of the multiple files switch conditions (one of the "Next file every ..." values) will switch to the next file. This will be a newly created file if value of "Ring buffer with n files" is not reached, otherwise it will replace the oldest of the formerly used files (thus forming a "ring").

This mode will limit the maximum disk usage, even for an unlimited amount of capture input data, keeping the latest captured data.

4.8. Link-layer header type

In the usual case, you won't have to choose this link-layer header type. The following paragraphs describe the exceptional cases, where selecting this type is possible, so you will have a guide of what to do:

If you are capturing on an 802.11 device on some versions of BSD, this might offer a choice of "Ethernet" or "802.11". "Ethernet" will cause the captured packets to have fake Ethernet headers; "802.11" will cause them to have IEEE 802.11 headers. Unless the capture needs to be read by an application that doesn't support 802.11 headers, you should select "802.11".

If you are capturing on an Endace DAG card connected to a synchronous serial line, this might offer a choice of "PPP over serial" or "Cisco HDLC"; if the protocol on the serial line is PPP, select "PPP over serial", and if the protocol on the serial line is Cisco HDLC, select "Cisco HDLC".

If you are capturing on an Endace DAG card connected to an ATM network, this might offer a choice of "RFC 1483 IP-over-ATM" or "Sun raw ATM". If the only traffic being captured is RFC 1483 LLC-encapsulated IP, or if the capture needs to be read by an application that doesn't support SunATM headers, select "RFC 1483 IP-over-ATM", otherwise select "Sun raw ATM".

If you are capturing on an Ethernet device, this might offer a choice of "Ethernet" or "DOCSIS". If you are capturing traffic from a Cisco Cable Modem Termination System that is putting DOCSIS traffic onto the Ethernet to be captured, select "DOCSIS", otherwise select "Ethernet".

4.9. Filtering while capturing

Wireshark uses the libpcap filter language for capture filters. This is explained in the tcpdump man page, which can be hard to understand, so it's explained here to some extent.

Tip!

You will find a lot of Capture Filter examples at http://wiki.wireshark.org/CaptureFilters.

You enter the capture filter into the Filter field of the Wireshark Capture Options dialog box, as shown in Figure 4.3, “The "Capture Options" dialog box”. The following is an outline of the syntax of the **tcpdump** capture filter language. See the expression option at the tcpdump manual page for details: http://www.tcpdump.org/tcpdump_man.html.

A capture filter takes the form of a series of primitive expressions connected by conjunctions (**and/or**) and optionally preceded by **not**:

```
[not] primitive [and|or [not] primitive ...]
```

An example is shown in Example 4.1, “ A capture filter for telnet that captures traffic to and from a particular host ”.

Example 4.1. A capture filter for telnet that captures traffic to and from a particular host

```
tcp port 23 and host 10.0.0.5
```

This example captures telnet traffic to and from the host 10.0.0.5, and shows how to use two primitives and the **and** conjunction. Another example is shown in Example 4.2, “ Capturing all telnet traffic not from 10.0.0.5”, and shows how to capture all telnet traffic except that from 10.0.0.5.

Example 4.2. Capturing all telnet traffic not from 10.0.0.5

```
tcp port 23 and not src host 10.0.0.5
```

XXX - add examples to the following list.

A primitive is simply one of the following:

[src|dst] host <host> — This primitive allows you to filter on a host IP address or name. You can optionally precede the primitive with the keyword **src|dst** to specify that you are only interested in source or destination addresses. If these are not present, packets where the specified address appears as either the source or the destination address will be selected.

ether [src|dst] host <ehost>	This primitive allows you to filter on Ethernet host addresses. You can optionally include the keyword **src|dst** between the keywords **ether** and **host** to specify that you are only interested in source or destination addresses. If these are not present, packets where the specified address appears in either the source or destination address will be selected.
gateway host <host>	This primitive allows you to filter on packets that used **host** as a gateway. That is, where the Ethernet source or destination was **host** but neither the source nor destination IP address was **host**.
[src|dst] net <net> [{mask <mask>}|{len <len>}]	This primitive allows you to filter on network numbers. You can optionally precede this primitive with the keyword **src|dst** to specify that you are only interested in a source or destination network. If neither of these are present, packets will be selected that have the specified network in either the source or destination address. In addition, you can specify either the netmask or the CIDR prefix for the network if they are different from your own.
[tcp|udp] [src|dst] port <port>	This primitive allows you to filter on TCP and UDP port numbers. You can optionally precede this primitive with the keywords **src|dst** and **tcp|udp** which allow you to specify that you are only interested in source or destination ports and TCP or UDP packets respectively. The keywords **tcp|udp** must appear before **src|dst**. If these are not specified, packets will be selected for both the TCP and UDP protocols and when the specified address appears in either the source or destination port field.
less|greater <length>	This primitive allows you to filter on packets whose length was less than or equal to the specified length, or greater than or equal to the specified length, respectively.
ip|ether proto <protocol>	This primitive allows you to filter on the specified protocol at either the Ethernet layer or the IP layer.
ether|ip broadcast|multicast	This primitive allows you to filter on either Ethernet or IP broadcasts or multicasts.
<expr> relop <expr>	This primitive allows you to create complex filter expressions that select bytes or ranges of bytes in packets. Please see the tcpdump man page at http:/ / www.tcpdump.org/ tcpdump_man.html for more details.

4.9.1. Automatic Remote Traffic Filtering

If Wireshark is running remotely (using e.g. SSH, an exported X11 window, a terminal server, ...), the remote content has to be transported over the network, adding a lot of (usually unimportant) packets to the actually interesting traffic.

To avoid this, Wireshark tries to figure out if it's remotely connected (by looking at some specific environment variables) and automatically creates a capture filter that matches aspects of the connection.

The following environment variables are analyzed:

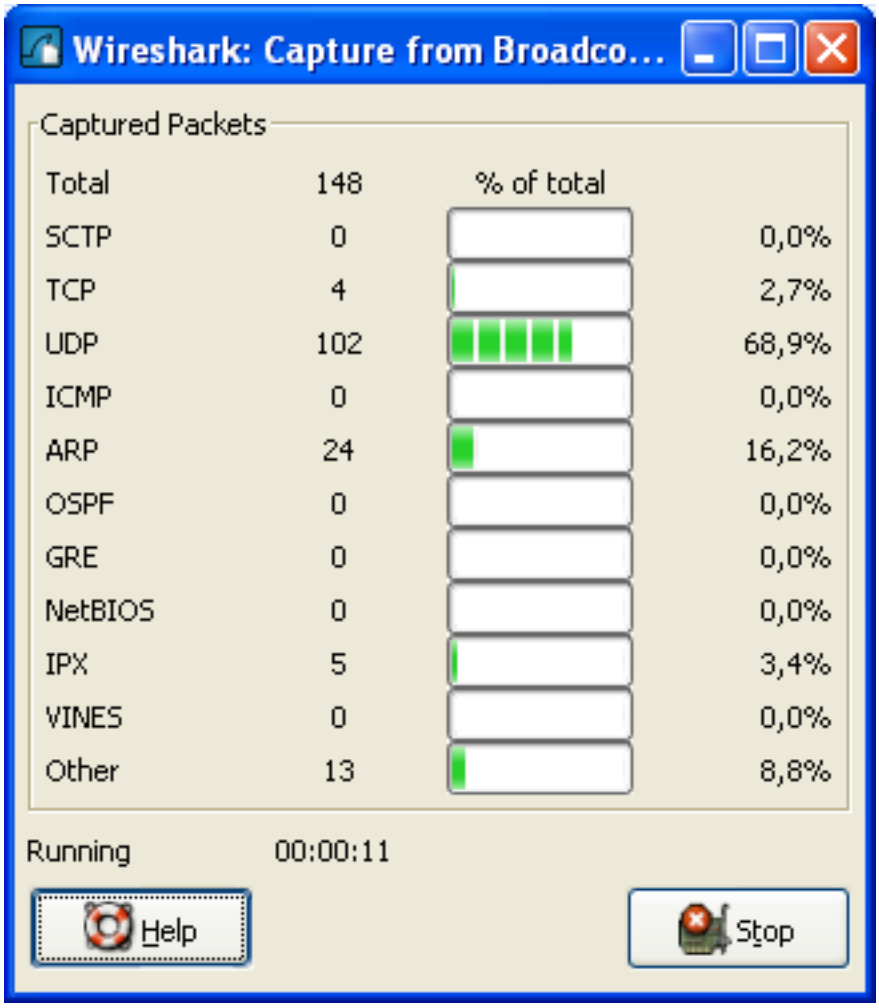
SSH_CONNECTION (ssh)	<remote IP> <remote port> <local IP> <local port>

SSH_CLIENT (ssh)	<remote IP> <remote port> <local port>
REMOTEHOST (tcsh, others?)	<remote name>
DISPLAY (x11)	[remote name]:<display num>
SESSIONNAME (terminal server)	<remote name>

4.10. While a Capture is running ...

While a capture is running, the following dialog box is shown:

Figure 4.5. The "Capture Info" dialog box



This dialog box will inform you about the number of captured packets and the time since the capture was started. The selection of which protocols are counted cannot be changed.

Tip!

This Capture Info dialog box can be hidden, using the "Hide capture info dialog" option in the Capture Options dialog box.

4.10.1. Stop the running capture

A running capture session will be stopped in one of the following ways:

1. Using the " Stop" button from the **Capture Info dialog box** .

Note!

The Capture Info dialog box might be hidden, if the option "Hide capture info dialog" is used.

2. Using the **menu item** "Capture/ Stop".
3. Using the **toolbar item** " Stop".
4. Pressing the accelerator keys: **Ctrl+E**.
5. The capture will be automatically stopped, if one of the **Stop Conditions** is exceeded, e.g. the maximum amount of data was captured.

4.10.2. Restart a running capture

A running capture session can be restarted with the same capture options as the last time, this will remove all packets previously captured. This can be useful, if some uninteresting packets are captured and there's no need to keep them.

Restart is a convenience function and equivalent to a capture stop following by an immediate capture start. A restart can be triggered in one of the following ways:

1. Using the **menu item** "Capture/ Restart".
2. Using the **toolbar item** " Restart".

Chapter 5. File Input / Output and Printing

5.1. Introduction

This chapter will describe input and output of capture data.

- Open/Import capture files in various capture file formats
- Save/Export capture files in various capture file formats
- Merge capture files together
- Print packets

5.2. Open capture files

Wireshark can read in previously saved capture files. To read them, simply select the menu or toolbar item: "File/ **Open**". Wireshark will then pop up the File Open dialog box, which is discussed in more detail in Section 5.2.1, “The "Open Capture File" dialog box”.

It's convenient to use drag-and-drop!

... to open a file, by simply dragging the desired file from your file manager and dropping it onto Wireshark's main window. However, drag-and-drop is not available/won't work in all desktop environments.

If you haven't previously saved the current capture file, you will be asked to do so, to prevent data loss (this behaviour can be disabled in the preferences).

In addition to its native file format (libpcap format, also used by tcpdump/WinDump and other libpcap/WinPcap-based programs), Wireshark can read capture files from a large number of other packet capture programs as well. See Section 5.2.2, “Input File Formats” for the list of capture formats Wireshark understands.

5.2.1. The "Open Capture File" dialog box

The "Open Capture File" dialog box allows you to search for a capture file containing previously captured packets for display in Wireshark. Table 5.1, “The system specific "Open Capture File" dialog box” shows some examples of the Wireshark Open File Dialog box.

The dialog appearance depends on your system!

The appearance of this dialog depends on the system and/or GTK+ toolkit version used. However, the functionality remains basically the same on any particular system.

Common dialog behaviour on all systems:

- Select files and directories.
- Click the Open/Ok button to accept your selected file and open it.
- Click the Cancel button to go back to Wireshark and not load a capture file.

Wireshark extensions to the standard behaviour of these dialogs:

- View file preview information (like the filesize, the number of packets, ...), if you've selected a capture file.
- Specify a display filter with the "Filter:" button and filter field. This filter will be used when opening the new file. The text field background becomes green for a valid filter string and red for an invalid one. Clicking on the Filter button causes Wireshark to pop up the Filters dialog box (which is discussed further in Section 6.3, “Filtering packets while viewing”).

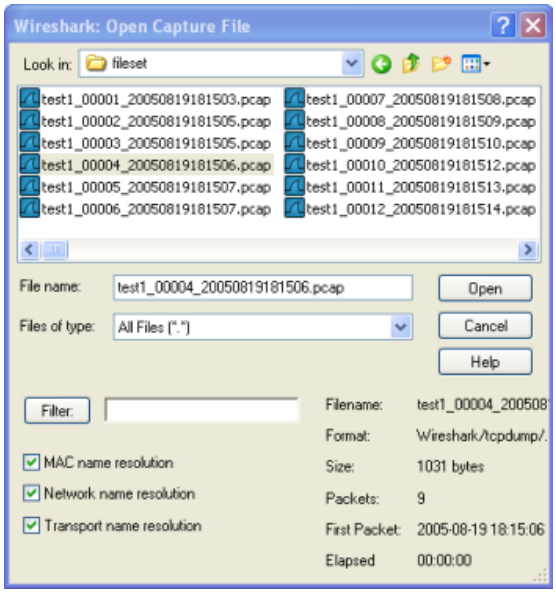
  XXX - we need a better description of these read filters

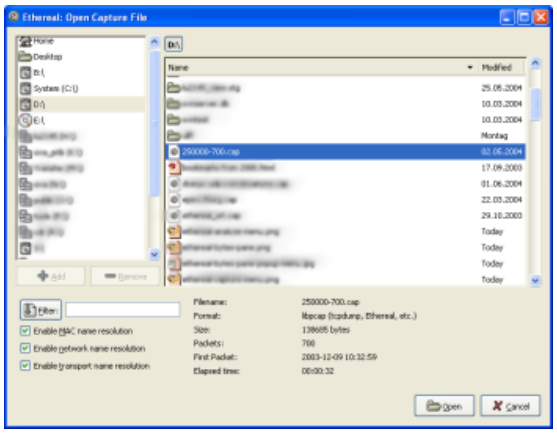
- Specify which type of name resolution is to be performed for all packets by clicking on one of the "... name resolution" check buttons. Details about name resolution can be found in Section 7.7, “Name Resolution”.

Save a lot of time loading huge capture files!

You can change the display filter and name resolution settings later while viewing the packets. However, loading huge capture files can take a significant amount of extra time if these settings are changed later, so in such situations it can be a good idea to set at least the filter in advance here.

Table 5.1. The system specific "Open Capture File" dialog box

Figure 5.1. "Open" on native Windows 	**Microsoft Windows** This is the common Windows file open dialog - plus some Wireshark extensions. Specific for this dialog: • If available, the "Help" button will lead you to this section of this "User's Guide". • XXX - the "Filter:" button currently doesn't work on Windows! • XXX - missing feature: If Wireshark doesn't recognize the selected file as a capture file, it should grey out the "Open" button.
Figure 5.2. "Open" - new GTK version	**Unix/Linux: GTK version >= 2.4** This is the common Gimp/GNOME file open dialog - plus some Wireshark extensions. Specific for this dialog: • The "+ Add" button allows you to add a directory, selected in the right-hand pane, to the favorites list on the left. Those changes are persistent. • The "- Remove" button allows you to remove a selected directory from that list again (the



items like: "Home", "Desktop", and "Filesystem" cannot be removed).

- If Wireshark doesn't recognize the selected file as a capture file, it will grey out the "Open" button.

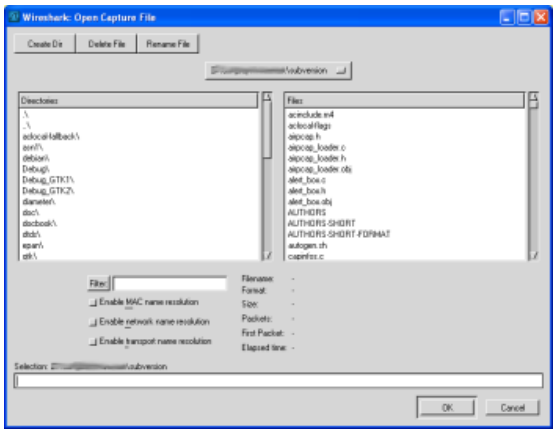
Figure 5.3. "Open" - old GTK version

Unix/Linux: GTK version < 2.4

This is the file open dialog of former Gimp/GNOME versions - plus some Wireshark extensions.

Specific for this dialog:

- If Wireshark doesn't recognize the selected file as a capture file, it will grey out the "Ok" button.



5.2.2. Input File Formats

The following file formats from other capture tools can be opened by Wireshark:

- libpcap, tcpdump and various other tools using tcpdump's capture format
- Sun snoop and atmsnoop
- Shomiti/Finisar *Surveyor* captures
- Novell *LANalyzer* captures
- Microsoft Network Monitor captures
- AIX's iptrace captures
- Cinco Networks NetXray captures
- Network Associates Windows-based Sniffer and Sniffer Pro captures

- Network General/Network Associates DOS-based Sniffer (compressed or uncompressed) captures
- AG Group/WildPackets EtherPeek/TokenPeek/AiroPeek/EtherHelp/PacketGrabber captures
- RADCOM's WAN/LAN Analyzer captures
- Network Instruments Observer version 9 captures
- Lucent/Ascend router debug output
- HP-UX's nettl
- Toshiba's ISDN routers dump output
- ISDN4BSD *i4btrace* utility
- traces from the EyeSDN USB S0
- IPLog format from the Cisco Secure Intrusion Detection System
- pppd logs (pppdump format)
- the output from VMS's TCPIPtrace/TCPtrace/UCX$TRACE utilities
- the text output from the DBS Etherwatch VMS utility
- Visual Networks' Visual UpTime traffic capture
- the output from CoSine L2 debug
- the output from Accellent's 5Views LAN agents
- Endace Measurement Systems' ERF format captures
- Linux Bluez Bluetooth stack hcidump -w traces
- Catapult DCT2000 .out files
- Gammu generated text output from Nokia DCT3 phones in Netmonitor mode
- IBM Series (OS/400) Comm traces (ASCII & UNICODE)
- Juniper Netscreen snoop captures
- Symbian OS btsnoop captures
- Tamosoft CommView captures
- Textronix K12xx 32bit .rf5 format captures
- Textronix K12 text file format captures
- Wireshark .pcapng captures (Experimental)
- ... new file formats are added from time to time

Opening a file may fail due to invalid packet types!

It may not be possible to read some formats dependent on the packet types captured. Ether-

net captures are usually supported for most file formats but it may not be possible to read other packet types (e.g. token ring packets) from all file formats.

5.3. Saving captured packets

You can save captured packets simply by using the Save As... menu item from the File menu under Wireshark. You can choose which packets to save and which file format to be used.

Saving may reduce the available information!

Saving the captured packets will slightly reduce the amount of information, e.g. the number of dropped packets will be lost; see Section A.1, “Capture Files” for details.

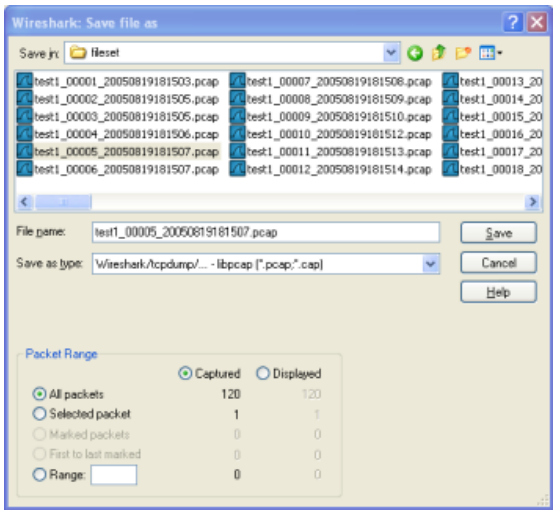
5.3.1. The "Save Capture File As" dialog box

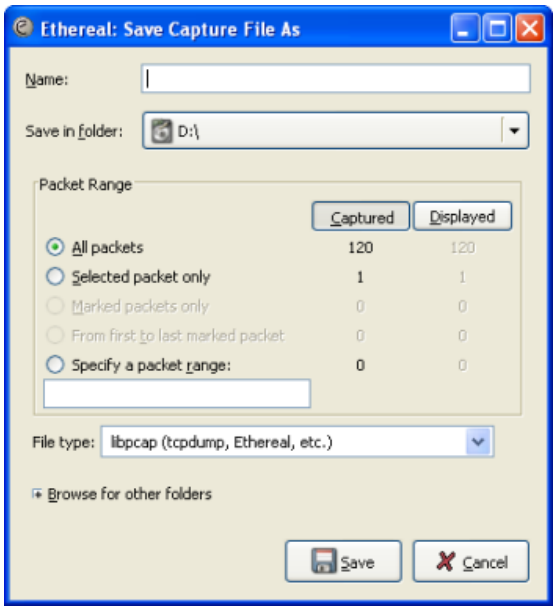
The "Save Capture File As" dialog box allows you to save the current capture to a file. Table 5.2, “The system specific "Save Capture File As" dialog box” shows some examples of this dialog box.

The dialog appearance depends on your system!

The appearance of this dialog depends on the system and GTK+ toolkit version used. However, the functionality remains basically the same on any particular system.

Table 5.2. The system specific "Save Capture File As" dialog box

Figure 5.4. "Save" on native Windows 	**Microsoft Windows** This is the common Windows file save dialog - plus some Wireshark extensions. Specific for this dialog: • If available, the "Help" button will lead you to this section of this "User's Guide". • If you don't provide a file extension to the filename - e.g. .pcap, Wireshark will append the standard file extension for that file format.
Figure 5.5. "Save" - new GTK version	**Unix/Linux: GTK version >= 2.4** This is the common Gimp/GNOME file save dialog - plus some Wireshark extensions. Specific for this dialog:

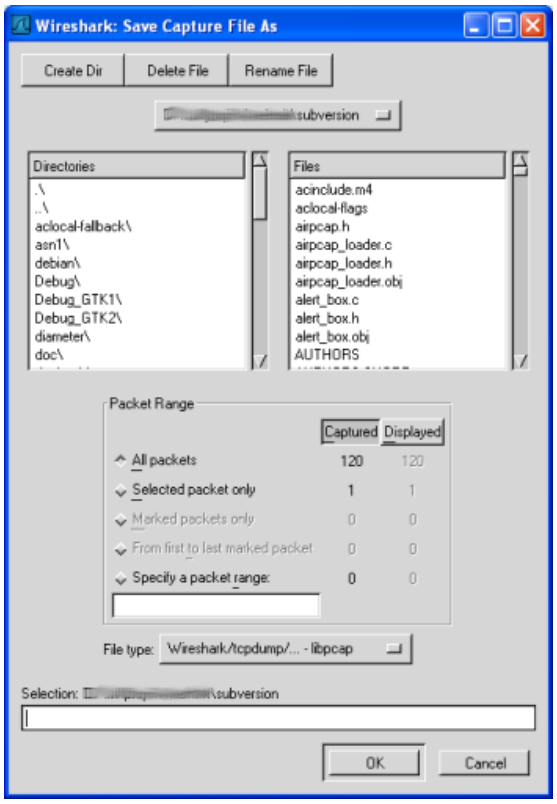


- Clicking on the + at "Browse for other folders" will allow you to browse files and folders in your file system.

Figure 5.6. "Save" - old GTK version

Unix/Linux: GTK version < 2.4

This is the file save dialog of former Gimp/GNOME versions - plus some Wireshark extensions.



With this dialog box, you can perform the following actions:

1. Type in the name of the file you wish to save the captured packets in, as a standard file name in your file system.

2. Select the directory to save the file into.

3. Select the range of the packets to be saved, see Section 5.8, “The Packet Range frame”

4. Specify the format of the saved capture file by clicking on the File type drop down box. You can choose from the types, described in Section 5.3.2, “Output File Formats”.

The selection of capture formats may be reduced!

Some capture formats may not be available, depending on the packet types captured.

File formats can be converted!

You can convert capture files from one format to another by reading in a capture file and writing it out using a different format.

5. Click on the Save/Ok button to accept your selected file and save to it. If Wireshark has a problem saving the captured packets to the file you specified, it will display an error dialog box. After clicking OK on that error dialog box, you can try again.

6. Click on the Cancel button to go back to Wireshark and not save the captured packets.

5.3.2. Output File Formats

Wireshark can save the packet data in its "native" file format (libpcap) and in the file formats of some other protocol analyzers, so other tools can read the capture data.

File formats have different time stamp accuracies!

Saving from the currently used file format to a different format may reduce the time stamp accuracy; see the Section 7.4, “Time Stamps” for details.

The following file formats can be saved by Wireshark (with the known file extensions):

- libpcap, tcpdump and various other tools using tcpdump's capture format (*.pcap,*.cap,*.dmp)
- Accellent 5Views (*.5vw)
- HP-UX's nettl (*.TRC0,*.TRC1)
- Microsoft Network Monitor - NetMon (*.cap)
- Network Associates Sniffer - DOS (*.cap,*.enc,*.trc,*fdc,*.syc)
- Network Associates Sniffer - Windows (*.cap)
- Network Instruments Observer version 9 (*.bfr)
- Novell LANalyzer (*.tr1)

- Sun snoop (*.snoop,*.cap)
- Visual Networks Visual UpTime traffic (*.*)
- ... new file formats are added from time to time

If the above tools will be more helpful than Wireshark is a different question ;-)

Third party protocol analyzers may require specific file extensions!

Other protocol analyzers than Wireshark may require that the file has a certain file extension in order to read the files you generate with Wireshark, e.g.:

".cap" for Network Associates Sniffer - Windows

5.4. Merging capture files

Sometimes you need to merge several capture files into one. For example this can be useful, if you have captured simultaneously from multiple interfaces at once (e.g. using multiple instances of Wireshark).

Merging capture files can be done in three ways:

- Use the **menu item "Merge"** from the "File" menu, to open the merge dialog, see Section 5.4.1, “The "Merge with Capture File" dialog box”. This menu item will be disabled, until you have loaded a capture file.
- Use **drag-and-drop** to drop multiple files on the main window. Wireshark will try to merge the packets in chronological order from the dropped files into a newly created temporary file. If you drop only a single file, it will simply replace a (maybe) existing one.
- Use the **mergecap** tool, which is a command line tool to merge capture files. This tool provides the most options to merge capture files, see Section D.7, “**mergecap**: Merging multiple capture files into one ”.

5.4.1. The "Merge with Capture File" dialog box

This dialog box let you select a file to be merged into the currently loaded file.

You will be prompted for an unsaved file first!

If your current data wasn't saved before, you will be asked to save it first, before this dialog box is shown.

Most controls of this dialog will work the same way as described in the "Open Capture File" dialog box, see Section 5.2.1, “The "Open Capture File" dialog box”.

Specific controls of this merge dialog are:

Prepend packets to existing file	Prepend the packets from the selected file before the currently loaded packets.
Merge packets chronologically	Merge both the packets from the selected and currently loaded file in chronological order.
Append packets to existing file	Append the packets from the selected file after the currently loaded packets.

Table 5.3. The system specific "Merge Capture File As" dialog box

Figure 5.7. "Merge" on native Windows	**Microsoft Windows** This is the common Windows file open dialog - plus some Wireshark extensions.

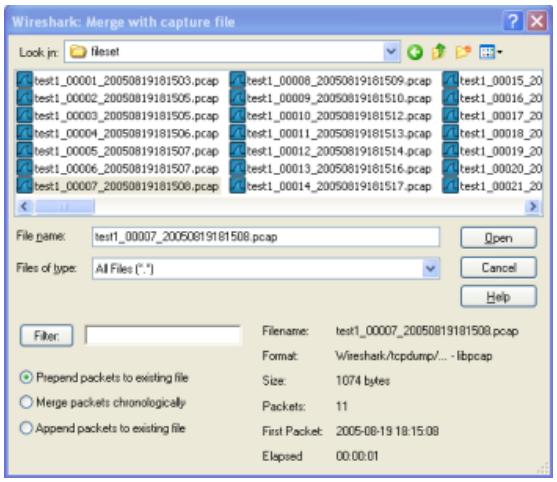


Figure 5.8. "Merge" - new GTK version

Unix/Linux: GTK version >= 2.4

This is the common Gimp/GNOME file open dialog - plus some Wireshark extensions.

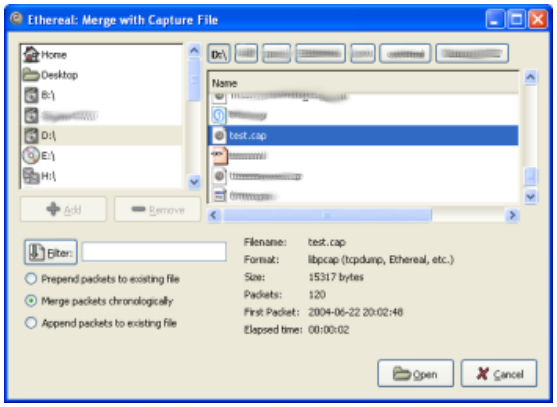
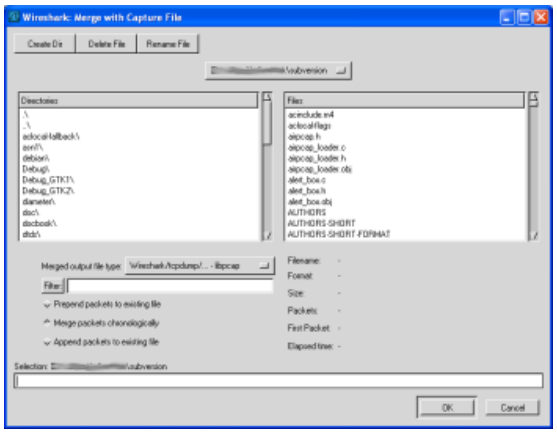


Figure 5.9. "Merge" - old GTK version

Unix/Linux: GTK version < 2.4

This is the file open dialog of former Gimp/GNOME versions - plus some Wireshark extensions.



5.5. File Sets

When using the "Multiple Files" option while doing a capture (see: Section 4.7, “Capture files and file modes”), the capture data is spread over several capture files, called a file set.

As it can become tedious to work with a file set by hand, Wireshark provides some features to handle these file sets in a convenient way.

How does Wireshark detect the files of a file set?

A filename in a file set uses the format Prefix_Number_DateTimeSuffix which might look like this: "test_00001_20060420183910.pcap". All files of a file set share the same prefix (e.g. "test") and suffix (e.g. ".pcap") and a varying middle part.

To find the files of a file set, Wireshark scans the directory where the currently loaded file resides and checks for files matching the filename pattern (prefix and suffix) of the currently loaded file.

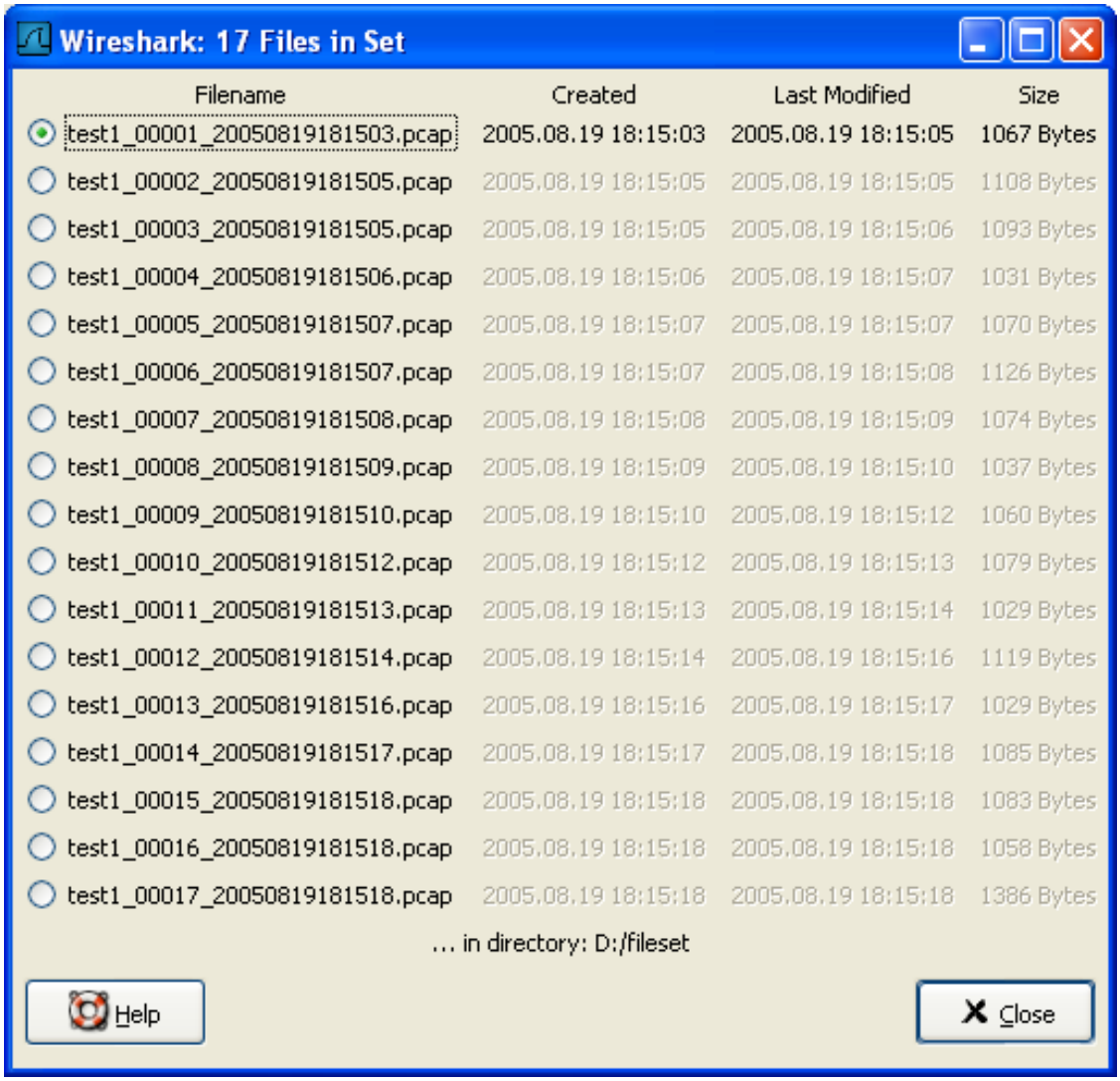
This simple mechanism usually works well, but has its drawbacks. If several file sets were captured with the same prefix and suffix, Wireshark will detect them as a single file set. If files were renamed or spread over several directories the mechanism will fail to find all files of a set.

The following features in the "File Set" submenu of the "File" menu are available to work with file sets in a convenient way:

- The **List Files** dialog box will list the files Wireshark has recognized as being part of the current file set.
- **Next File** closes the current and opens the next file in the file set.
- **Previous File** closes the current and opens the previous file in the file set.

5.5.1. The "List Files" dialog box

Figure 5.10. The "List Files" dialog box



Each line contains information about a file of the file set:

- **Filename** the name of the file. If you click on the filename (or the radio button left to it), the current file will be closed and the corresponding capture file will be opened.
- **Created** the creation time of the file
- **Last Modified** the last time the file was modified
- **Size** the size of the file

The last line will contain info about the currently used directory where all of the files in the file set can be found.

The content of this dialog box is updated each time a capture file is opened/closed.

The Close button will, well, close the dialog box.

5.6. Exporting data

Wireshark provides several ways and formats to export packet data. This section describes general ways to export data from Wireshark.

Note!

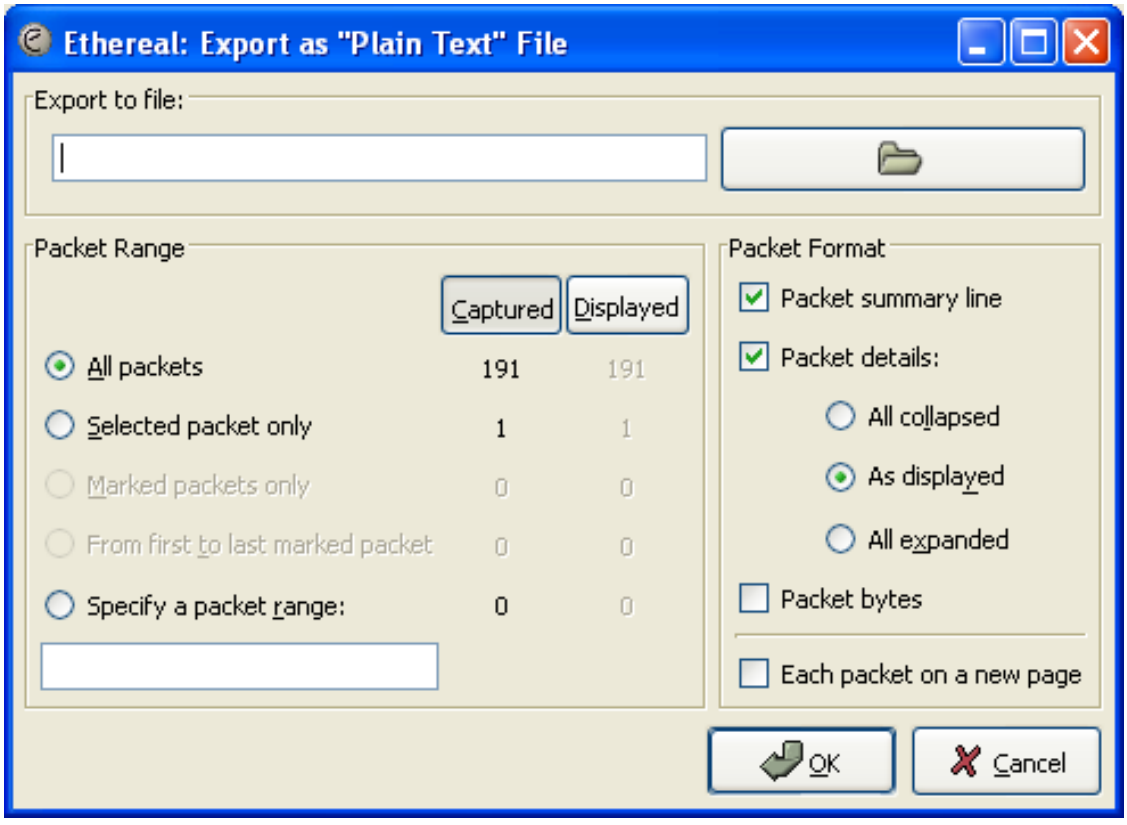
There are more specialized functions to export specific data, which will be described at the appropriate places.

XXX - add detailed descriptions of the output formats and some sample output, too.

5.6.1. The "Export as Plain Text File" dialog box

Export packet data into a plain ASCII text file, much like the format used to print packets.

Figure 5.11. The "Export as Plain Text File" dialog box



- **Export to file:** frame chooses the file to export the packet data to.
- The **Packet Range** frame is described in Section 5.8, “The Packet Range frame”.
- The **Packet Details** frame is described in Section 5.9, “The Packet Format frame”.

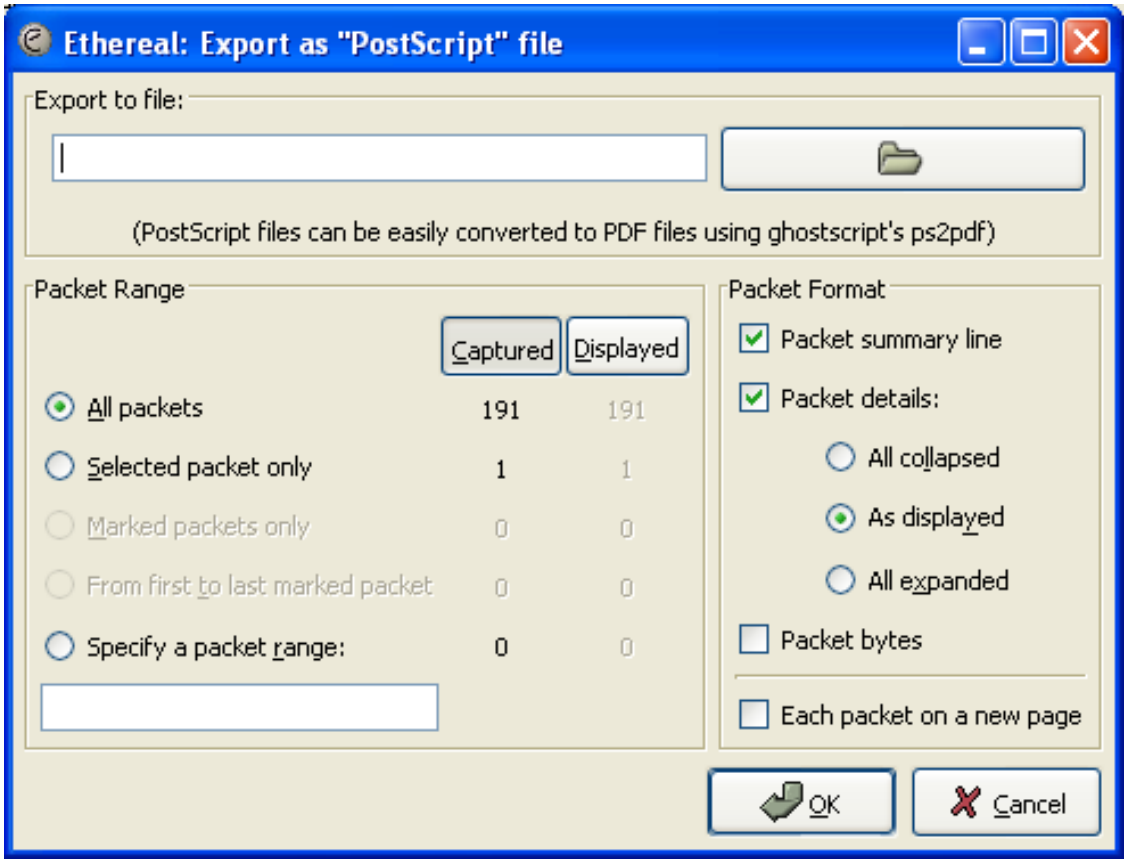
5.6.2. The "Export as PostScript File" dialog box

Export packet data into PostScript, much like the format used to print packets.

Tip!

You can easily convert PostScript files to PDF files using ghostscript. For example: export to a file named foo.ps and then call: **ps2pdf foo.ps**

Figure 5.12. The "Export as PostScript File" dialog box



- **Export to file:** frame chooses the file to export the packet data to.
- The **Packet Range** frame is described in Section 5.8, “The Packet Range frame”.
- The **Packet Details** frame is described in Section 5.9, “The Packet Format frame”.

5.6.3. The "Export as CSV (Comma Separated Values) File" dialog box

XXX - add screenshot

Export packet summary into CSV, used e.g. by spreadsheet programs to im-/export data.

- **Export to file:** frame chooses the file to export the packet data to.
- The **Packet Range** frame is described in Section 5.8, “The Packet Range frame”.

5.6.4. The "Export as C Arrays (packet bytes) file" dialog box

XXX - add screenshot

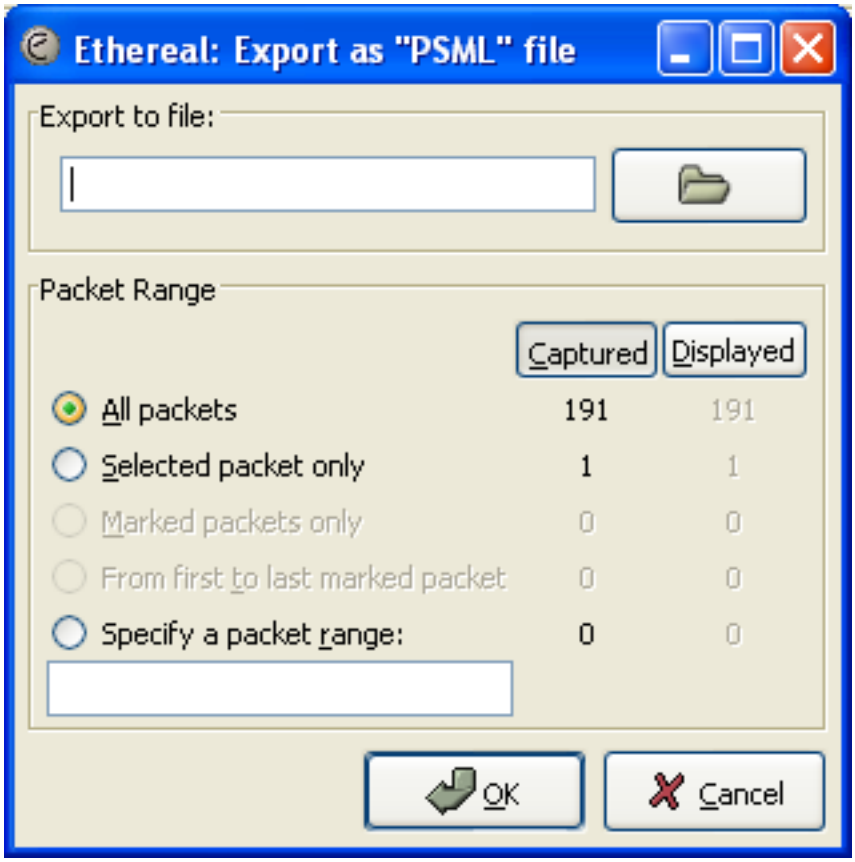
Export packet bytes into C arrays so you can import the stream data into your own C program.

- **Export to file:** frame chooses the file to export the packet data to.
- The **Packet Range** frame is described in Section 5.8, “The Packet Range frame”.

5.6.5. The "Export as PSML File" dialog box

Export packet data into PSML. This is an XML based format including only the packet summary. The PSML file specification is available at: http://www.nbee.org/doku.php?id=netpdl:psml_specification.

Figure 5.13. The "Export as PSML File" dialog box



- **Export to file:** frame chooses the file to export the packet data to.
- The **Packet Range** frame is described in Section 5.8, “The Packet Range frame”.

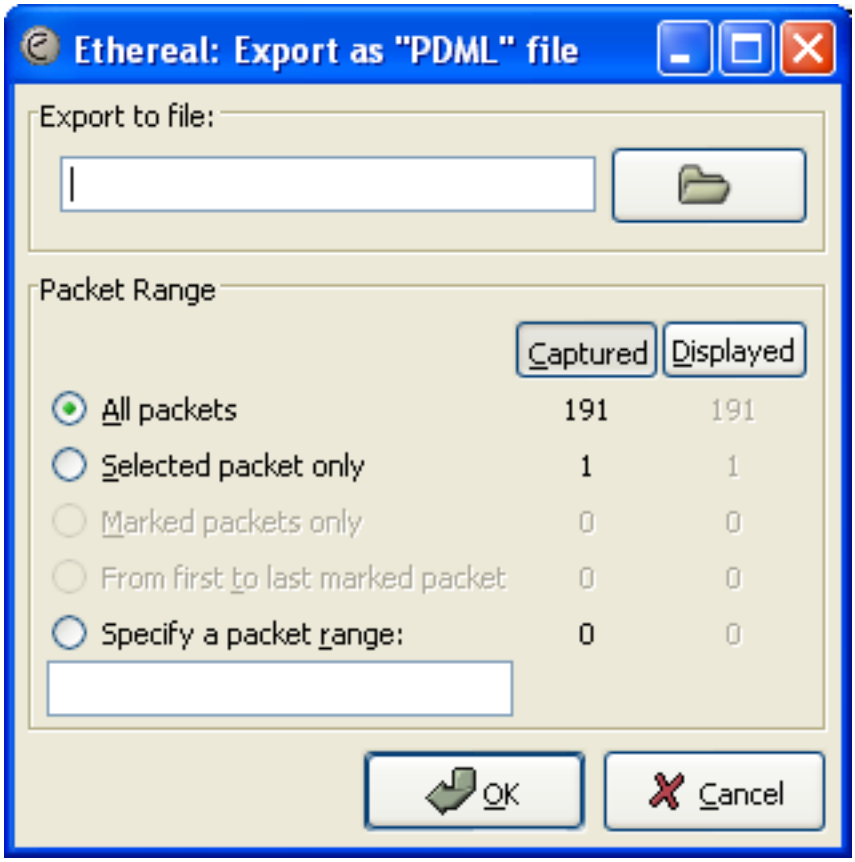
There's no such thing as a packet details frame for PSML export, as the packet format is defined by the PSML specification.

5.6.6. The "Export as PDML File" dialog box

Export packet data into PDML. This is an XML based format including the packet details. The PDML file specification is available at: http://www.nbee.org/doku.php?id=netpdl:pdml_specification.

The PDML specification is not officially released and Wireshark's implementation of it is still in an early beta state, so please expect changes in future Wireshark versions.

Figure 5.14. The "Export as PDML File" dialog box



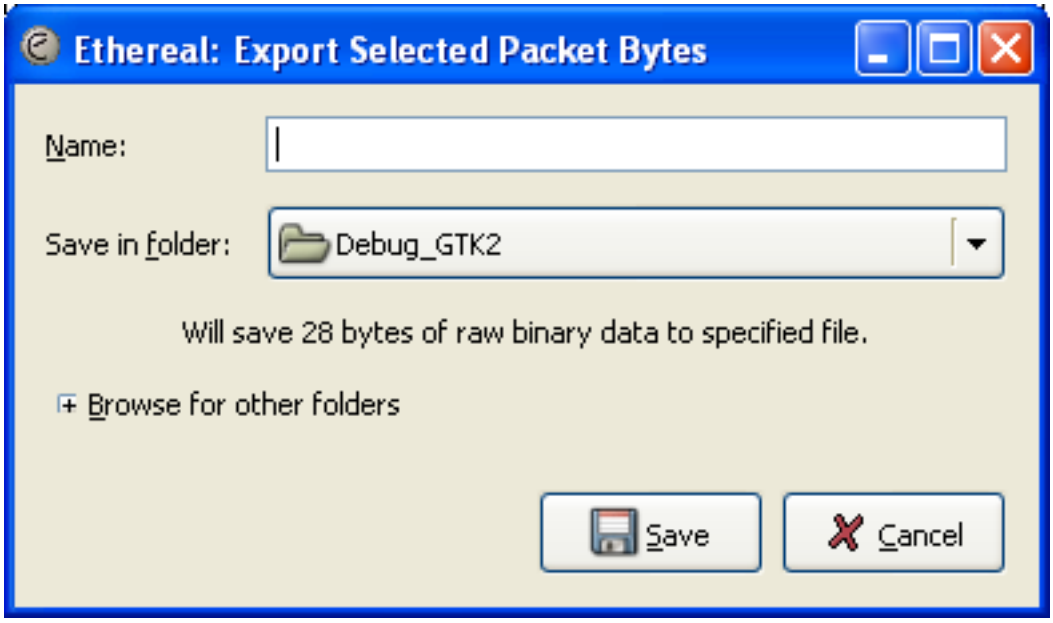
- **Export to file:** frame chooses the file to export the packet data to.
- The **Packet Range** frame is described in Section 5.8, “The Packet Range frame”.

There's no such thing as a packet details frame for PDML export, as the packet format is defined by the PDML specification.

5.6.7. The "Export selected packet bytes" dialog box

Export the bytes selected in the "Packet Bytes" pane into a raw binary file.

Figure 5.15. The "Export Selected Packet Bytes" dialog box

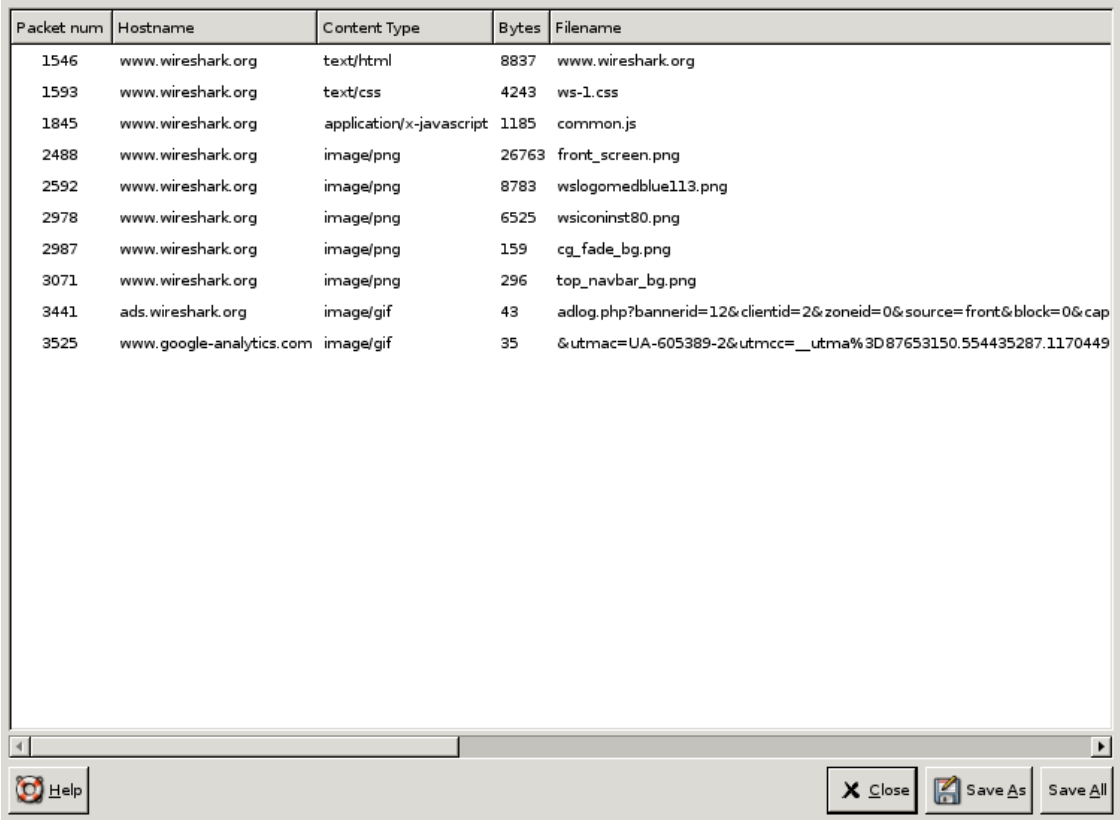


- **Name:** the filename to export the packet data to.
- The **Save in folder:** field lets you select the folder to save to (from some predefined folders).
- **Browse for other folders** provides a flexible way to choose a folder.

5.6.8. The "Export Objects" dialog box

This feature scans through HTTP streams in the currently open capture file or running capture and takes reassembled objects such as HTML documents, image files, executables and anything else that can be transferred over HTTP and lets you save them to disk. If you have a capture running, this list is automatically updated every few seconds with any new objects seen. The saved objects can then be opened with the proper viewer or executed in the case of executables (if it is for the same platform you are running Wireshark on) without any further work on your part. This feature is not available when using GTK2 versions below 2.4.

Figure 5.16. The "Export Objects" dialog box



Columns:

- **Packet num:** The packet number in which this object was found. In some cases, there can be multiple objects in the same packet.
- **Hostname:** The hostname of the server that sent the object as a response to an HTTP request.
- **Content Type:** The HTTP content type of this object.
- **Bytes:** The size of this object in bytes.
- **Filename:** The final part of the URI (after the last slash). This is typically a filename, but may be a long complex looking string, which typically indicates that the file was received in response to a HTTP POST request.

Buttons:

- **Help:** Opens this section in the user's guide.
- **Close:** Closes this dialog.
- **Save As:** Saves the currently selected object as a filename you specify. The default filename to save as is taken from the filename column of the objects list.
- **Save All:** Saves all objects in the list using the filename from the filename column. You will be asked what directory / folder to save them in. If the filename is invalid for the operating system / file system you are running Wireshark on, then an error will appear and that object will not be saved (but

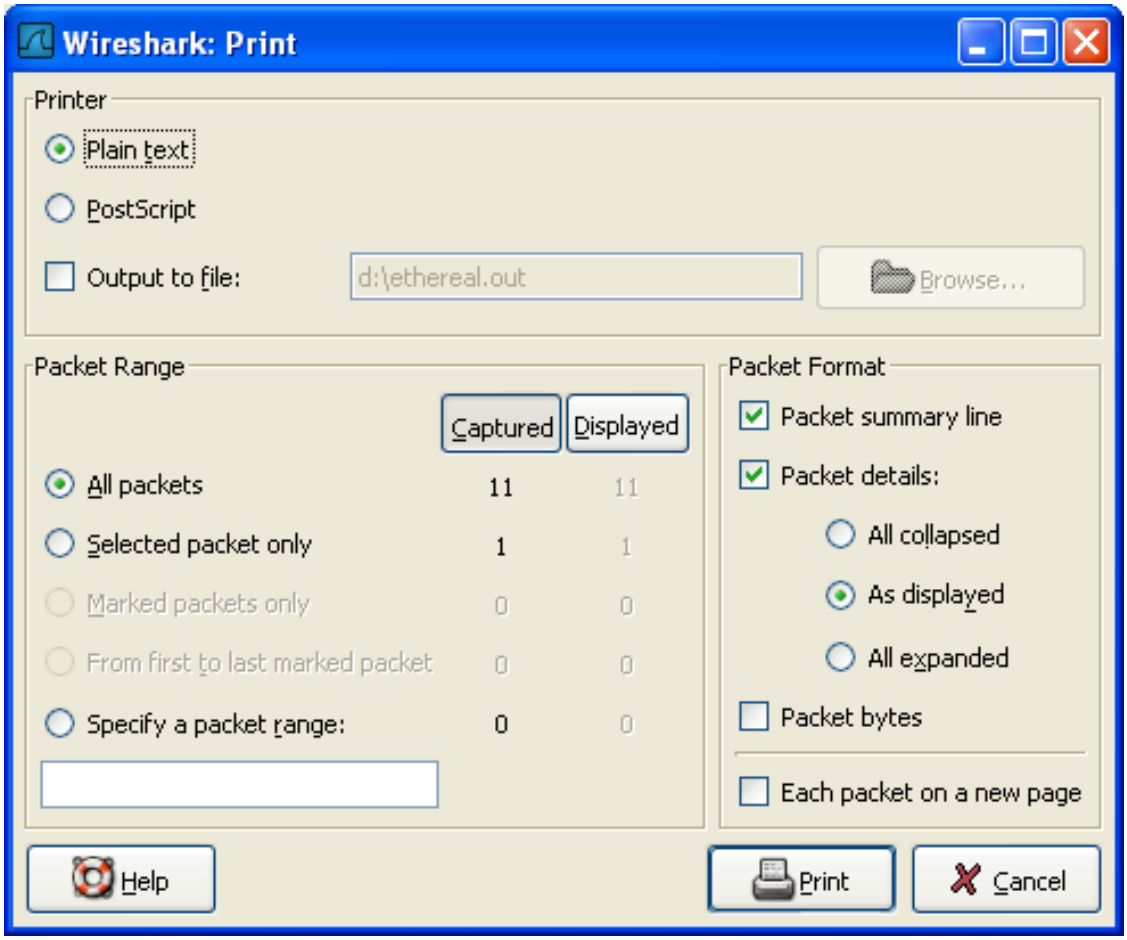
all of the others will be).

5.7. Printing packets

To print packets, select the "Print..." menu item from the File menu. When you do this, Wireshark pops up the Print dialog box as shown in Figure 5.17, “The "Print" dialog box”.

5.7.1. The "Print" dialog box

Figure 5.17. The "Print" dialog box



The following fields are available in the Print dialog box:

Printer This field contains a pair of mutually exclusive radio buttons:

- **Plain Text** specifies that the packet print should be in plain text.
- **PostScript** specifies that the packet print process should use PostScript to generate a better print output on PostScript aware printers.
- **Output to file:** specifies that printing be done to a file, using the filename entered in the field or selected with the browse button.

This field is where you enter the **file** to print to if you have selected Print to a file, or you can click the button to browse the filesystem. It is greyed out if Print to a file is not selected.

- **Print command** specifies that a command be used for printing.

Note!

These **Print command** fields are not available on windows platforms.

This field specifies the command to use for printing. It is typically **lpr**. You would change it to specify a particular queue if you need to print to a queue other than the default. An example might be:

```
lpr -Pmypostscript
```

This field is greyed out if **Output to file:** is checked above.

Packet Range Select the packets to be printed, see Section 5.8, “The Packet Range frame”

Packet Format Select the output format of the packets to be printed. You can choose, how each packet is printed, see Figure 5.19, “The "Packet Format" frame”

5.8. The Packet Range frame

The packet range frame is a part of various output related dialog boxes. It provides options to select which packets should be processed by the output function.

Figure 5.18. The "Packet Range" frame

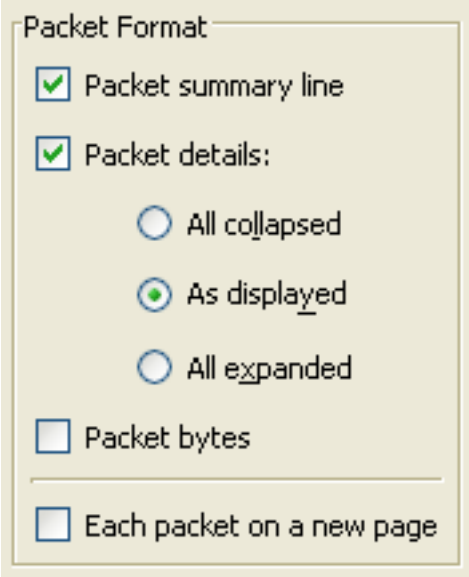
If the **Captured** button is set (default), all packets from the selected rule will be processed. If the **Displayed** button is set, only the currently displayed packets are taken into account to the selected rule.

- **All packets** will process all packets.
- **Selected packet only** process only the selected packet.
- **Marked packets only** process only the marked packets.
- **From first to last marked packet** process the packets from the first to the last marked one.
- **Specify a packet range** process a user specified range of packets, e.g. specifying **5,10-15,20-** will process the packet number five, the packets from packet number ten to fifteen (inclusive) and every packet from number twenty to the end of the capture.

5.9. The Packet Format frame

The packet format frame is a part of various output related dialog boxes. It provides options to select which parts of a packet should be used for the output function.

Figure 5.19. The "Packet Format" frame



- **Packet summary line** enable the output of the summary line, just as in the "Packet List" pane.
- **Packet details** enable the output of the packet details tree.
  - **All collapsed** the info from the "Packet Details" pane in "all collapsed" state.
  - **As displayed** the info from the "Packet Details" pane in the current state.
  - **All expanded** the info from the "Packet Details" pane in "all expanded" state.
- **Packet bytes** enable the output of the packet bytes, just as in the "Packet Bytes" pane.
- **Each packet on a new page** put each packet on a separate page (e.g. when saving/printing to a text file, this will put a form feed character between the packets).

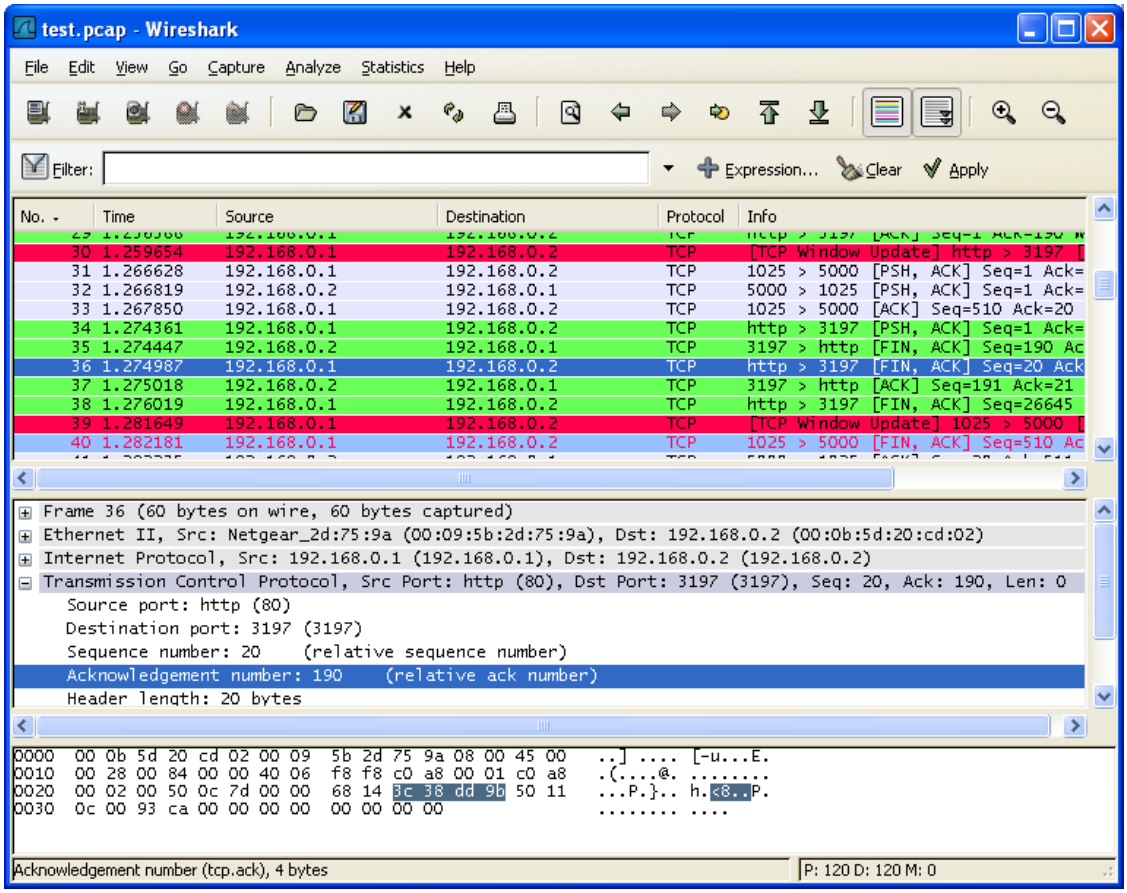
Chapter 6. Working with captured packets

6.1. Viewing packets you have captured

Once you have captured some packets, or you have opened a previously saved capture file, you can view the packets that are displayed in the packet list pane by simply clicking on a packet in the packet list pane, which will bring up the selected packet in the tree view and byte view panes.

You can then expand any part of the tree view by clicking on the **plus** sign (the symbol itself may vary) to the left of that part of the payload, and you can select individual fields by clicking on them in the tree view pane. An example with a TCP packet selected is shown in Figure 6.1, “Wireshark with a TCP packet selected for viewing”. It also has the Acknowledgment number in the TCP header selected, which shows up in the byte view as the selected bytes.

Figure 6.1. Wireshark with a TCP packet selected for viewing

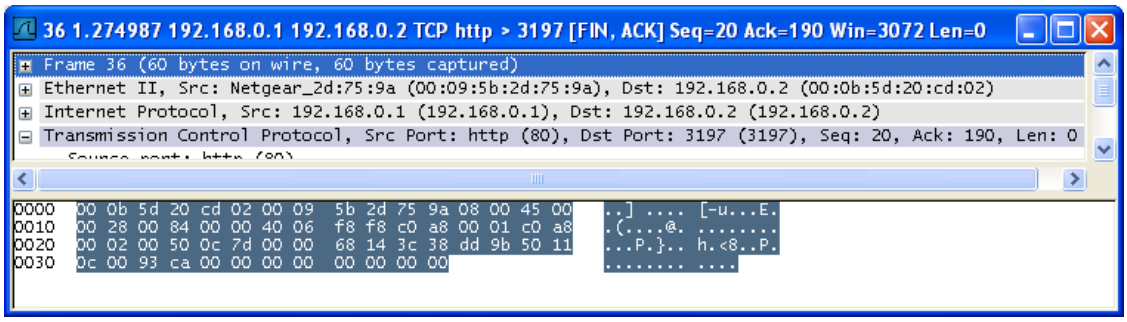

test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
30 1.259654 192.168.0.1 192.168.0.2 TCP [TCP Window Update] http > 3197 [
31 1.266628 192.168.0.1 192.168.0.2 TCP 1025 > 5000 [PSH, ACK] Seq=1 Ack=
32 1.266819 192.168.0.2 192.168.0.1 TCP 5000 > 1025 [PSH, ACK] Seq=1 Ack=
33 1.267850 192.168.0.1 192.168.0.2 TCP 1025 > 5000 [ACK] Seq=510 Ack=20
34 1.274361 192.168.0.1 192.168.0.2 TCP http > 3197 [PSH, ACK] Seq=1 Ack=
35 1.274447 192.168.0.2 192.168.0.1 TCP 3197 > http [FIN, ACK] Seq=190 Ac
36 1.274987 192.168.0.1 192.168.0.2 TCP http > 3197 [FIN, ACK] Seq=20 Ack
37 1.275018 192.168.0.2 192.168.0.1 TCP 3197 > http [ACK] Seq=191 Ack=21
38 1.276019 192.168.0.1 192.168.0.2 TCP http > 3197 [FIN, ACK] Seq=26645
39 1.281649 192.168.0.1 192.168.0.2 TCP [TCP Window Update] 1025 > 5000 [
40 1.282181 192.168.0.1 192.168.0.2 TCP 1025 > 5000 [FIN, ACK] Seq=510 Ac
Frame 36 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: Netgear_2d:75:9a (00:09:5b:2d:75:9a), Dst: 192.168.0.2 (00:0b:5d:20:cd:02)
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.2 (192.168.0.2)
Transmission Control Protocol, Src Port: http (80), Dst Port: 3197 (3197), Seq: 20, Ack: 190, Len: 0
Source port: http (80)
Destination port: 3197 (3197)
Sequence number: 20 (relative sequence number)
Acknowledgement number: 190 (relative ack number)
Header length: 20 bytes
0000 00 0b 5d 20 cd 02 00 09 5b 2d 75 9a 08 00 45 00 ..] .... [-u...E.
0010 00 28 00 84 00 00 40 06 f8 f8 c0 a8 00 01 c0 a8 .(....@. ....
0020 00 02 00 50 0c 7d 00 00 68 14 8c 38 dd 9b 50 11 ...P.}.. h..8..P.
0030 0c 00 93 ca 00 00 00 00 00 00 00 00 ....
Acknowledgement number (tcp.ack), 4 bytes
P: 120 D: 120 M: 0

You can also select and view packets the same way, while Wireshark is capturing, if you selected "Update list of packets in real time" in the Wireshark Capture Preferences dialog box.

In addition, you can view individual packets in a separate window as shown in Figure 6.2, “Viewing a

packet in a separate window". Do this by selecting the packet in which you are interested in the packet list pane, and then select "Show Packet in New Windows" from the Display menu. This allows you to easily compare two or even more packets.

Figure 6.2. Viewing a packet in a separate window

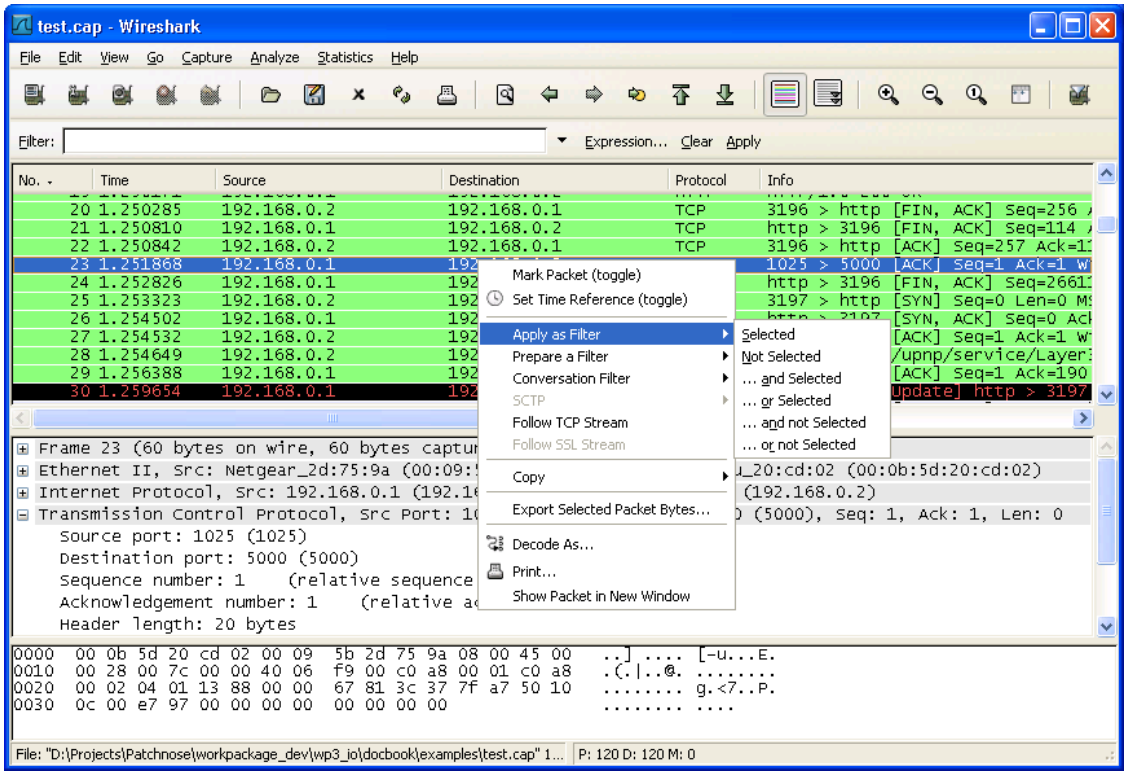


6.2. Pop-up menus

You can bring up a pop-up menu over either the "Packet List" or "Packet Details" pane by clicking your right mouse button at the corresponding pane.

6.2.1. Pop-up menu of the "Packet List" pane

Figure 6.3. Pop-up menu of the "Packet List" pane


test.cap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
20 1.250285 192.168.0.2 192.168.0.1 TCP 3196 > http [FIN, ACK] Seq=256
21 1.250810 192.168.0.1 192.168.0.2 TCP http > 3196 [FIN, ACK] Seq=114
22 1.250842 192.168.0.2 192.168.0.1 TCP 3196 > http [ACK] Seq=257 Ack=1
23 1.251868 192.168.0.1 192 1025 > 5000 [ACK] Seq=1 Ack=1 W
24 1.252826 192.168.0.1 192 http > 3196 [FIN, ACK] Seq=2661
25 1.253323 192.168.0.2 192 3197 > http [SYN] Seq=0 Len=0 M
26 1.254502 192.168.0.1 192 [SYN, ACK] Seq=0 Ac
27 1.254532 192.168.0.2 192 [ACK] Seq=1 Ack=1 W
28 1.254649 192.168.0.2 192 /upnp/service/Layer
29 1.256388 192.168.0.1 192 [ACK] Seq=1 Ack=190
30 1.259654 192.168.0.1 192 Update] http > 3197
Mark Packet (toggle)
Set Time Reference (toggle)
Apply as Filter
Prepare a Filter
Conversation Filter
SCTP
Follow TCP Stream
Follow SSL Stream
Copy
Export Selected Packet Bytes...
Decode As...
Print...
Show Packet in New Window
Selected
Not Selected
... and Selected
... or Selected
... and not Selected
... or not Selected
Frame 23 (60 bytes on wire, 60 bytes captu
Ethernet II, Src: Netgear_2d:75:9a (00:09:
Internet Protocol, Src: 192.168.0.1 (192.16
Transmission Control Protocol, Src Port: 10
Source port: 1025 (1025)
Destination port: 5000 (5000)
Sequence number: 1 (relative sequence
Acknowledgement number: 1 (relative a
Header length: 20 bytes
0000 00 0b 5d 20 cd 02 00 09 5b 2d 75 9a 08 00 45 00
0010 00 28 00 7c 00 00 40 06 f9 00 c0 a8 00 01 c0 a8
0020 00 02 04 01 13 88 00 00 67 81 3c 37 7f a7 50 10
0030 0c 00 e7 97 00 00 00 00 00 00 00 00
File: "D:\Projects\Patchnose\workpackage_dev\wp3_io\docbook\examples\test.cap" 1...
P: 120 D: 120 M: 0

The following table gives an overview of which functions are available in this pane, where to find the corresponding function in the main menu, and a short description of each item.

Table 6.1. The menu items of the "Packet List" pop-up menu

Item	Identical to main menu's item:	Description
Mark Packet (toggle)	Edit	Mark/unmark a packet.
Set Time Reference (toggle)	Edit	Set/reset a time reference.

Apply as Filter	Analyze	Prepare and apply a display filter based on the currently selected item.

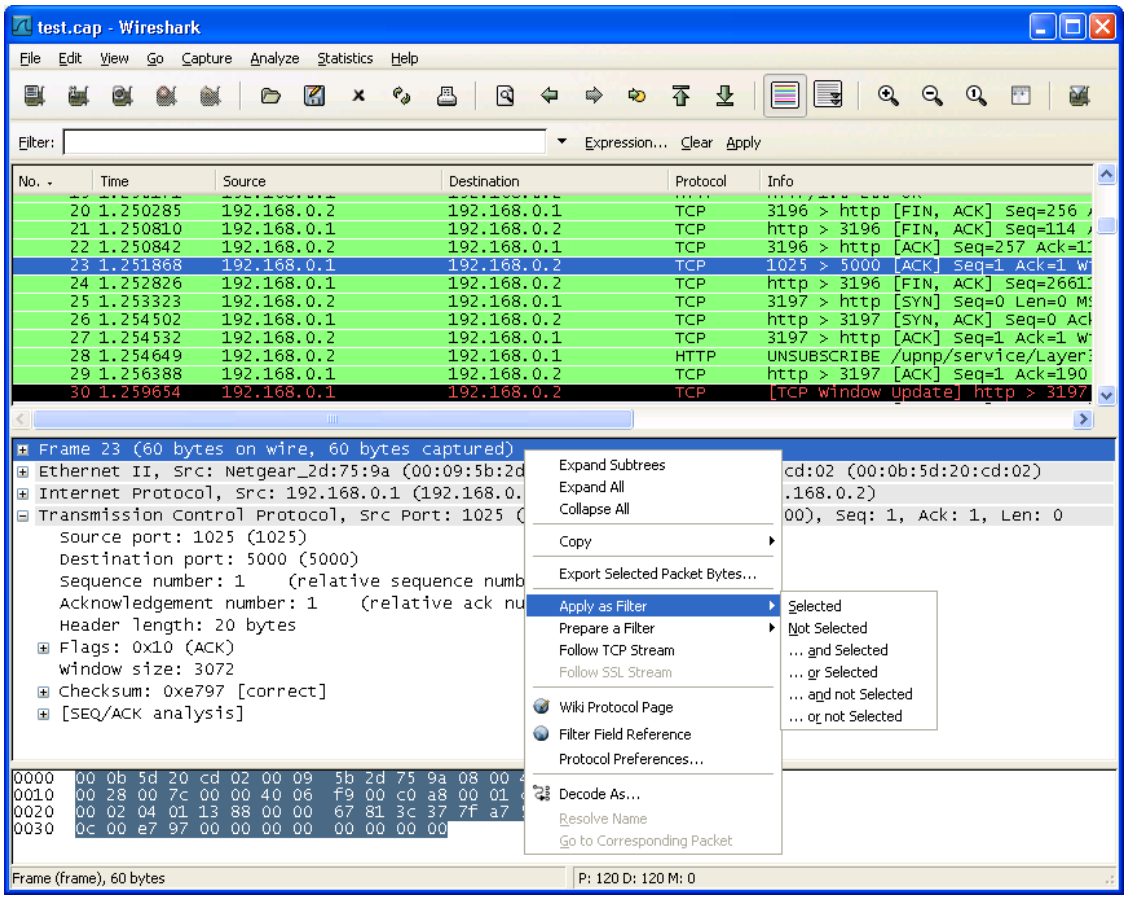
Item	Identical to main menu's item:	Description
Prepare a Filter	Analyze	Prepare a display filter based on the currently selected item.
Conversation Filter	-	This menu item applies a display filter with the address information from the selected packet. E.g. the IP menu entry will set a filter to show the traffic between the two IP addresses of the current packet. XXX - add a new section describing this better.
Colorize Conversation	-	This menu item uses a display filter with the address information from the selected packet to build a new colorizing rule.
SCTP	-	XXX - add an explanation of this.
Follow TCP Stream	Analyze	Allows you to view all the data on a TCP stream between a pair of nodes.
Follow SSL Stream	Analyze	Same as "Follow TCP Stream" but for SSL. XXX - add a new section describing this better.

Copy/ Summary (Text)	-	Copy the summary fields as displayed to the clipboard, as tab-separated text.
Copy/ Summary (CSV)	-	Copy the summary fields as displayed to the clipboard, as comma-separated text.
Copy/ As Filter		Prepare a display filter based on the currently selected item and copy that filter to the clipboard.
Copy/ Bytes (Offset Hex Text)	-	Copy the packet bytes to the clipboard in hexdump-like format.
Copy/ Bytes (Offset Hex)	-	Copy the packet bytes to the clipboard in hexdump-like format, but without the text portion.
Copy/ Bytes (Printable Text Only)	-	Copy the packet bytes to the clipboard as ASCII text, excluding non-printable characters.
Copy/ Bytes (Hex Stream)	-	Copy the packet bytes to the clipboard as an unpunctuated list of hex digits.
Copy/ Bytes (Binary Stream)	-	Copy the packet bytes to the clipboard as raw binary. The data is stored in the clipboard as MIME-type "application/octet-stream". This option is not available in versions of Wireshark built using GTK+ 1.x.
Export Selected Packet Bytes...	File	This menu item is the same as the File menu item of the same name. It allows you to export raw packet bytes to a binary file.

Item	Identical to main menu's item:	Description
Decode As...	Analyze	Change or apply a new relation between two dissectors.
Print...	File	Print packets.
Show Packet in New Window	View	Display the selected packet in a new window.

6.2.2. Pop-up menu of the "Packet Details" pane

Figure 6.4. Pop-up menu of the "Packet Details" pane


test.cap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
20 1.250285 192.168.0.2 192.168.0.1 TCP 3196 > http [FIN, ACK] Seq=256
21 1.250810 192.168.0.1 192.168.0.2 TCP http > 3196 [FIN, ACK] Seq=114
22 1.250842 192.168.0.2 192.168.0.1 TCP 3196 > http [ACK] Seq=257 Ack=1
23 1.251868 192.168.0.1 192.168.0.2 TCP 1025 > 5000 [ACK] Seq=1 Ack=1
24 1.252826 192.168.0.1 192.168.0.2 TCP http > 3196 [FIN, ACK] Seq=2661
25 1.253323 192.168.0.2 192.168.0.1 TCP 3197 > http [SYN] Seq=0 Len=0
26 1.254502 192.168.0.1 192.168.0.2 TCP http > 3197 [SYN, ACK] Seq=0
27 1.254532 192.168.0.2 192.168.0.1 TCP 3197 > http [ACK] Seq=1 Ack=1
28 1.254649 192.168.0.2 192.168.0.1 HTTP UNSUBSCRIBE /upnp/service/Layer
29 1.256388 192.168.0.1 192.168.0.2 TCP http > 3197 [ACK] Seq=1 Ack=190
30 1.259654 192.168.0.1 192.168.0.2 TCP [TCP Window Update] http > 3197
Frame 23 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: Netgear_2d:75:9a (00:09:5b:2d
Internet Protocol, Src: 192.168.0.1 (192.168.0.
Transmission Control Protocol, Src Port: 1025 (
Source port: 1025 (1025)
Destination port: 5000 (5000)
Sequence number: 1 (relative sequence numb
Acknowledgement number: 1 (relative ack nu
Header length: 20 bytes
Flags: 0x10 (ACK)
Window size: 3072
Checksum: 0xe797 [correct]
[SEQ/ACK analysis]
0000 00 0b 5d 20 cd 02 00 09 5b 2d 75 9a 08 00
0010 00 28 00 7c 00 00 40 06 f9 00 c0 a8 00 01
0020 00 02 04 01 13 88 00 00 67 81 3c 37 7f a7
0030 0c 00 e7 97 00 00 00 00 00 00 00 00
cd:02 (00:0b:5d:20:cd:02)
.168.0.2)
00), Seq: 1, Ack: 1, Len: 0
Expand Subtrees
Expand All
Collapse All
Copy
Export Selected Packet Bytes...
Apply as Filter
Prepare a Filter
Follow TCP Stream
Follow SSL Stream
Wiki Protocol Page
Filter Field Reference
Protocol Preferences...
Decode As...
Resolve Name
Go to Corresponding Packet
Selected
Not Selected
... and Selected
... or Selected
... and not Selected
... or not Selected
Frame (frame), 60 bytes
P: 120 D: 120 M: 0

The following table gives an overview of which functions are available in this pane, where to find the corresponding function in the main menu, and a short description of each item.

Table 6.2. The menu items of the "Packet Details" pop-up menu

Item	Identical to main menu's item:	Description
Expand Subtrees	View	Expand the currently selected subtree.
Expand All	View	Expand all subtrees in all packets in the capture.
Collapse All	View	Wireshark keeps a list of all the protocol subtrees that are expanded, and uses it to ensure that the correct subtrees are expanded when you display a packet. This menu item collapses the tree view of all packets in the capture list.

Copy/ Description	Edit	Copy the displayed text of the selected field to the system clipboard.
Copy/ Fieldname	Edit	Copy the name of the selected field to the system clipboard.
Copy/ Value	Edit	Copy the value of the selected field to the system clipboard.
Copy/ As Filter	Edit	Prepare a display filter based on the currently selected item and copy it to the clipboard.
Copy/ Bytes (Offset Hex Text)	-	Copy the packet bytes to the clipboard in hexdump-like format; similar to the Packet List Pane command, but copies only the bytes relevant to the selected part of the tree (the bytes selected in the Packet Bytes Pane).
Copy/ Bytes (Offset Hex)	-	Copy the packet bytes to the clipboard in hexdump-like format, but without the text portion; similar to the Packet List Pane command, but copies only the bytes relevant to the selected part of the tree (the bytes selected in the Packet Bytes Pane).
Copy/ Bytes (Printable Text Only)	-	Copy the packet bytes to the clipboard as ASCII text, excluding non-printable characters; similar to the Packet List Pane command, but copies only the bytes relevant to the selected part of the tree (the bytes selected in the Packet Bytes Pane).
Copy/ Bytes (Hex Stream)	-	Copy the packet bytes to the clipboard as an unpunctuated list of hex digits; similar to the Packet List Pane command, but copies only the bytes relevant to the selected part of the tree (the bytes selected in the Packet Bytes Pane).
Copy/ Bytes (Binary Stream)	-	Copy the packet bytes to the clipboard as raw binary; similar to the Packet List Pane command, but copies only the bytes relevant to the selected part of the tree (the bytes selected in the Packet Bytes Pane). The data is stored in the clipboard as MIME-type "application/octet-stream". This option is not available in versions of Wireshark built using GTK+ 1.x.
Export Selected Packet Bytes...	File	This menu item is the same as the File menu item of the same name. It allows you to export raw packet bytes to a binary file.

Item	Identical to main menu's item:	Description

Apply as Filter	Analyze	Prepare and apply a display filter based on the currently selected item.
Prepare a Filter	Analyze	Prepare a display filter based on the currently selected item.
Colorize with Filter	-	Prepare a display filter based on the currently selected item and use it to prepare a new colorize rule.
Follow TCP Stream	Analyze	Allows you to view all the data on a TCP stream between a pair of nodes.
Follow SSL Stream	Analyze	Same as "Follow TCP Stream" but for SSL. XXX - add a new section describing this better.

Wiki Protocol Page	-	Show the wiki page corresponding to the currently selected protocol in your web browser.
Filter Field Reference	-	Show the filter field reference web page corresponding to the currently selected protocol in your web browser.
Protocol Preferences...	-	The menu item takes you to the properties dialog and selects the page corresponding to the protocol if there are properties associated with the highlighted field. More information on preferences can be found in Figure 9.8, “The preferences dialog box”.

Decode As...	Analyze	Change or apply a new relation between two dissectors.
Resolve Name	View	Causes a name resolution to be performed for the selected packet, but NOT every packet in the capture.
Go to Corresponding Packet	Go	If the selected field has a corresponding packet, go to it. Corresponding packets will usually be a request/response packet pair or such.

6.3. Filtering packets while viewing

Wireshark has two filtering languages: One used when capturing packets, and one used when displaying packets. In this section we explore that second type of filter: Display filters. The first one has already been dealt with in Section 4.9, “Filtering while capturing”.

Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. They allow you to select packets by:

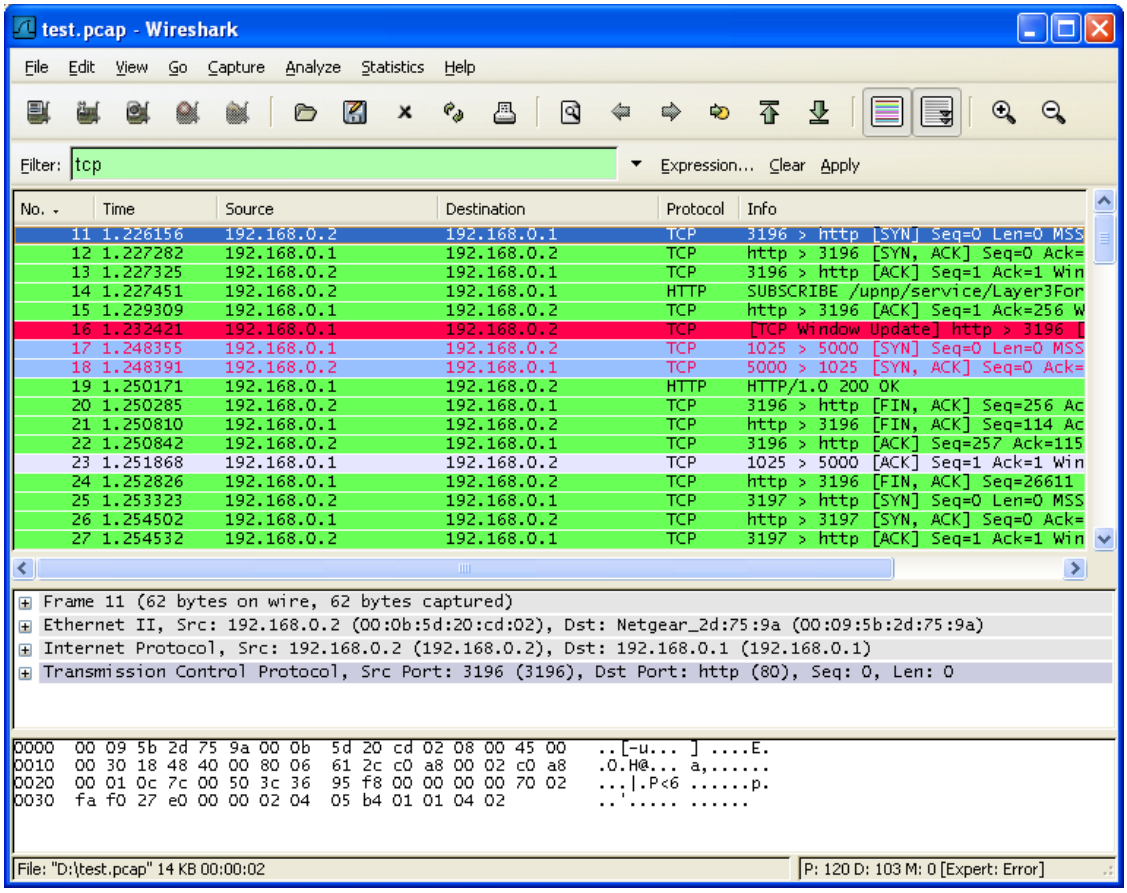
- Protocol
- The presence of a field
- The values of fields
- A comparison between fields
- ... and a lot more!

To select packets based on protocol type, simply type the protocol in which you are interested in the **Filter:** field in the filter toolbar of the Wireshark window and press enter to initiate the filter. Figure 6.5, “Filtering on the TCP protocol” shows an example of what happens when you type **tcp** in the filter field.

Note!

All protocol and field names are entered in lowercase. Also, don't forget to press enter after entering the filter expression.

Figure 6.5. Filtering on the TCP protocol



As you might have noticed, only packets of the TCP protocol are displayed now (e.g. packets 1-10 are hidden). The packet numbering will remain as before, so the first packet shown is now packet number 11.

Note!

When using a display filter, all packets remain in the capture file. The display filter only changes the display of the capture file but not its content!

You can filter on any protocol that Wireshark understands. You can also filter on any field that a dissector adds to the tree view, but only if the dissector has added an abbreviation for the field. A list of such fields is available in Wireshark in the **Add Expression...** dialog box. You can find more information on the **Add Expression...** dialog box in Section 6.5, “The "Filter Expression" dialog box”.

For example, to narrow the packet list pane down to only those packets to or from the IP address 192.168.0.1, use **ip.addr==192.168.0.1**.

Note!

To remove the filter, click on the **Clear** button to the right of the filter field.

6.4. Building display filter expressions

Wireshark provides a simple but powerful display filter language that allows you to build quite complex filter expressions. You can compare values in packets as well as combine expressions into more specific expressions. The following sections provide more information on doing this.

Tip!

You will find a lot of Display Filter examples at the **Wireshark Wiki Display Filter page** at http://wiki.wireshark.org/DisplayFilters.

6.4.1. Display filter fields

Every field in the packet details pane can be used as a filter string, this will result in showing only the packets where this field exists. For example: the filter string: **tcp** will show all packets containing the tcp protocol.

There is a complete list of all filter fields available through the menu item "Help/Supported Protocols" in the page "Display Filter Fields" of the Supported Protocols dialog.

XXX - add some more info here and a link to the statusbar info.

6.4.2. Comparing values

You can build display filters that compare values using a number of different comparison operators. They are shown in Table 6.3, “Display Filter comparison operators”.

Tip!

You can use English and C-like terms in the same way, they can even be mixed in a filter string!

Table 6.3. Display Filter comparison operators

English	C-like	Description and example
eq	==	**Equal** `ip.src==10.0.0.5`
ne	!=	**Not equal** `ip.src!=10.0.0.5`
gt	>	**Greater than** `frame.len > 10`

English	C-like	Description and example
lt	<	**Less than** `frame.len < 128`
ge	>=	**Greater than or equal to** `frame.len ge 0x100`
le	<=	**Less than or equal to** `frame.len <= 0x20`

In addition, all protocol fields are typed. Table 6.4, “Display Filter Field Types” provides a list of the types and example of how to express them.

Table 6.4. Display Filter Field Types

Type	Example

Unsigned integer (8-bit, 16-bit, 24-bit, 32-bit)	You can express integers in decimal, octal, or hexadecimal. The following display filters are equivalent:
	
	`ip.len le 1500`
	`ip.len le 02734`
	`ip.len le 0x436`

Signed integer (8-bit, 16-bit, 24-bit, 32-bit)	
Boolean	A boolean field is present in the protocol decode only if its value is true. For example, **tcp.flags.syn** is present, and thus true, only if the SYN flag is present in a TCP segment header. Thus the filter expression **tcp.flags.syn** will select only those packets for which this flag exists, that is, TCP segments where the segment header contains the SYN flag. Similarly, to find source-routed token ring packets, use a filter expression of **tr.sr**.

Ethernet address (6 bytes)	Separators can be a colon (:), dot (.) or dash (-) and can have one or two bytes between separators:
	
	`eth.dst == ff:ff:ff:ff:ff:ff`
	`eth.dst == ff-ff-ff-ff-ff-ff`
	`eth.dst == ffff.ffff.ffff`

IPv4 address	ip.addr == 192.168.0.1

Type	Example
	Classless InterDomain Routing (CIDR) notation can be used to test if an IPv4 address is in a certain subnet. For example, this display filter will find all packets in the 129.111 Class-B network: ip.addr == 129.111.0.0/16
IPv6 address	ipv6.addr == ::1
IPX address	ipx.addr == 00000000.ffffffffffff
String (text)	http.request.uri == "http://www.wireshark.org/"

6.4.3. Combining expressions

You can combine filter expressions in Wireshark using the logical operators shown in Table 6.5, "Display Filter Logical Operations"

Table 6.5. Display Filter Logical Operations

English	C-like	Description and example
and	&&	**Logical AND** `ip.src==10.0.0.5 and tcp.flags.fin`
or	||	**Logical OR** `ip.scr==10.0.0.5 or ip.src==192.1.1.1`
xor	^^	**Logical XOR** `tr.dst[0:3] == 0.6.29 xor tr.src[0:3] == 0.6.29`
not	!	**Logical NOT** `not llc`
[...]		**Substring Operator** Wireshark allows you to select subsequences of a sequence in rather elaborate ways. After a label you can place a pair of brackets [] containing a comma separated list of range specifiers. `eth.src[0:3] == 00:00:83` The example above uses the n:m format to specify a single range. In this case n is the beginning offset and m is the length of the range being specified.

English	C-like	Description and example

		`eth.src[1-2] == 00:83`
		
		The example above uses the n-m format to specify a single range. In this case n is the beginning offset and m is the ending offset.
		
		`eth.src[:4] == 00:00:83:00`
		
		The example above uses the :m format, which takes everything from the beginning of a sequence to offset m. It is equivalent to 0:m
		
		`eth.src[4:] == 20:20`
		
		The example above uses the n: format, which takes everything from offset n to the end of the sequence.
		
		`eth.src[2] == 83`
		
		The example above uses the n format to specify a single range. In this case the element in the sequence at offset n is selected. This is equivalent to n:1.
		
		`eth.src[0:3,1-2,:4,4:,2] == 00:00:83:00:83:00:00:83:00:20:20:83`
		
		Wireshark allows you to string together single ranges in a comma separated list to form compound ranges as shown above.

6.4.4. A common mistake

Warning!

Using the != operator on combined expressions like: eth.addr, ip.addr, tcp.port, udp.port and alike will probably not work as expected!

Often people use a filter string to display something like **ip.addr == 1.2.3.4** which will display all packets containing the IP address 1.2.3.4.

Then they use **ip.addr != 1.2.3.4** to see all packets not containing the IP address 1.2.3.4 in it. Unfortunately, this does **not** do the expected.

Instead, that expression will even be true for packets where either source or destination IP address equals 1.2.3.4. The reason for this, is that the expression **ip.addr != 1.2.3.4** must be read as "the packet contains a field named ip.addr with a value different from 1.2.3.4". As an IP datagram contains both a source and a destination address, the expression will evaluate to true whenever at least one of the two addresses differs from 1.2.3.4.

If you want to filter out all packets containing IP datagrams to or from IP address 1.2.3.4, then the correct filter is **!(ip.addr == 1.2.3.4)** as it reads "show me all the packets for which it is not true that a field

named ip.addr exists with a value of 1.2.3.4", or in other words, "filter out all packets for which there are no occurrences of a field named ip.addr with the value 1.2.3.4".

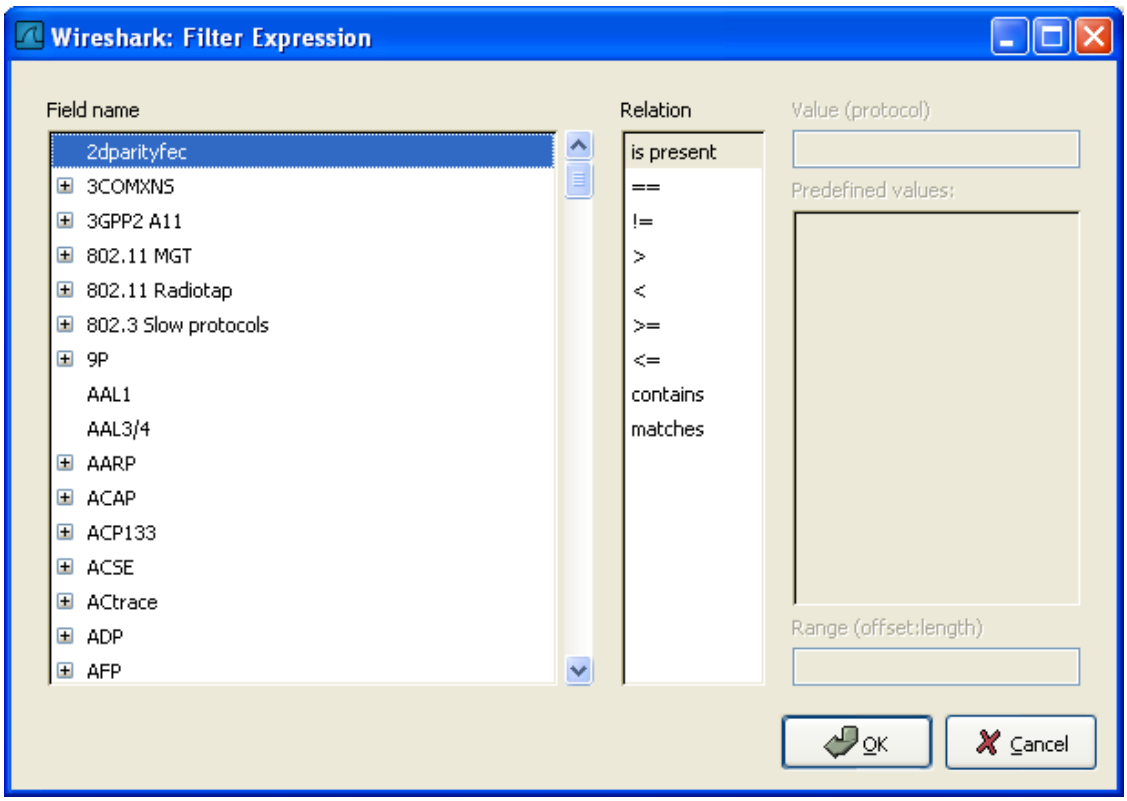
6.5. The "Filter Expression" dialog box

When you are accustomed to Wireshark's filtering system and know what labels you wish to use in your filters it can be very quick to simply type a filter string. However if you are new to Wireshark or are working with a slightly unfamiliar protocol it can be very confusing to try to figure out what to type. The Filter Expression dialog box helps with this.

Tip!

The "Filter Expression" dialog box is an excellent way to learn how to write Wireshark display filter strings.

Figure 6.6. The "Filter Expression" dialog box



When you first bring up the Filter Expression dialog box you are shown a tree list of field names, organized by protocol, and a box for selecting a relation.

Field Name Select a protocol field from the protocol field tree. Every protocol with filterable fields is listed at the top level. (You can search for a particular protocol entry by entering the first few letters of the protocol name). By clicking on the "+" next to a protocol name you can get a list of the field names available for filtering for that protocol.

Relation Select a relation from the list of available relation. The **is present** is a unary relation which is true if the selected field is present in a packet. All other listed relations are binary relations which require additional data (e.g. a **Value** to match) to complete.

When you select a field from the field name list and select a binary relation (such as the equality relation ==) you will be given the opportunity to enter a value, and possibly some range information.

Value — You may enter an appropriate value in the **Value** text box. The **Value** will also indicate the type of value for the **field name** you have selected (like character string).

Predefined values — Some of the protocol fields have predefined values available, much like enum's in C. If the selected protocol field has such values defined, you can choose one of them here.

Range — XXX - add an explanation here!

OK — When you have built a satisfactory expression click **OK** and a filter string will be built for you.

Cancel — You can leave the **Add Expression...** dialog box without any effect by clicking the **Cancel** button.

6.6. Defining and saving filters

You can define filters with Wireshark and give them labels for later use. This can save time in remembering and retyping some of the more complex filters you use.

To define a new filter or edit an existing one, select the **Capture Filters...** menu item from the Capture menu or the **Display Filters...** menu item from the Analyze menu. Wireshark will then pop up the Filters dialog as shown in Figure 6.7, “The "Capture Filters" and "Display Filters" dialog boxes”.

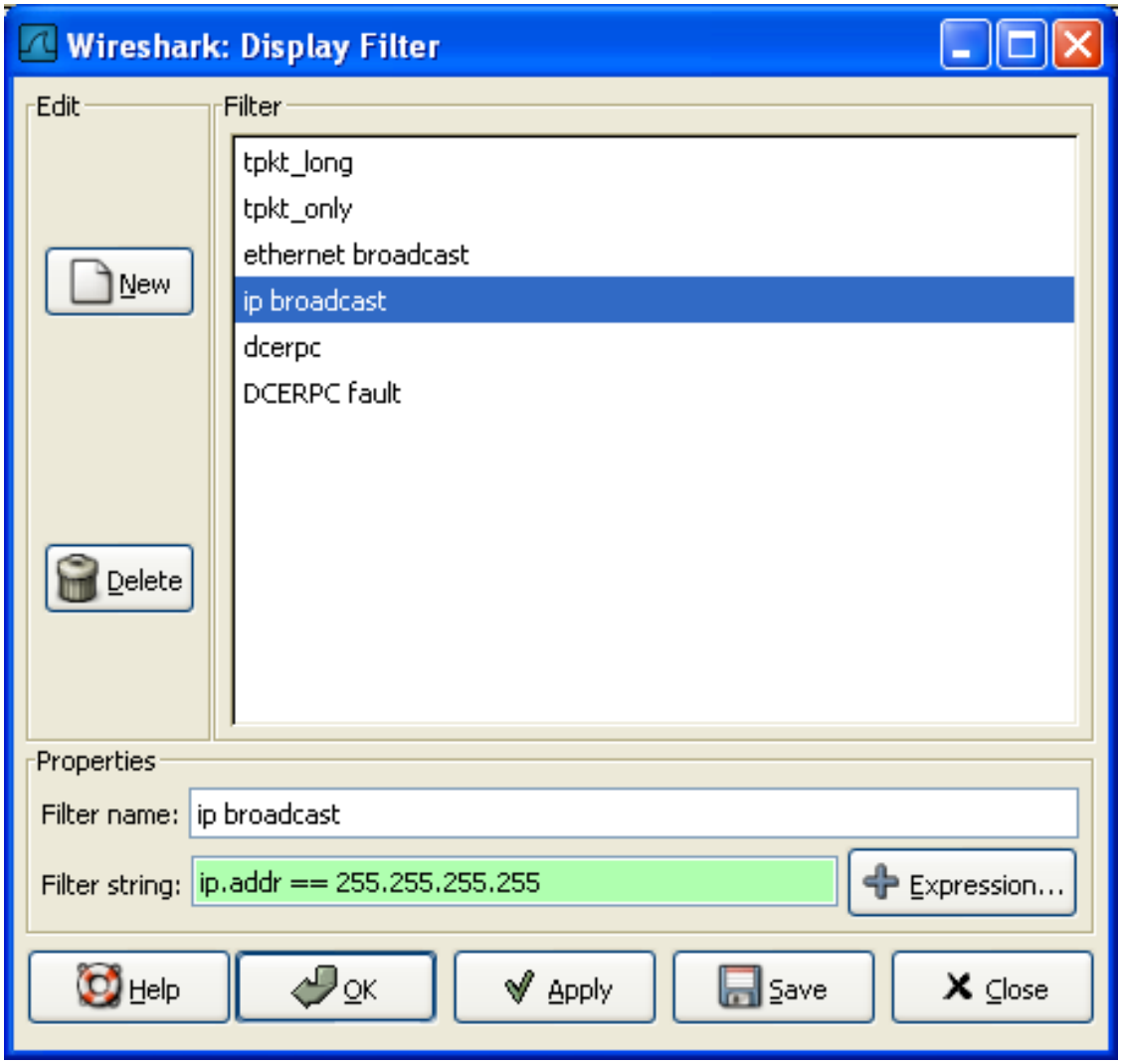
Note!

The mechanisms for defining and saving capture filters and display filters are almost identical. So both will be described here, differences between these two will be marked as such.

Warning!

You must use **Save** to save your filters permanently. **Ok** or **Apply** will not save the filters, so they will be lost when you close Wireshark.

Figure 6.7. The "Capture Filters" and "Display Filters" dialog boxes



New This button adds a new filter to the list of filters. The currently entered values from Filter name and Filter string will be used. If any of these fields are empty, it will be set to "new".

Delete This button deletes the selected filter. It will be greyed out, if no filter is selected.

Filter You can select a filter from this list (which will fill in the filter name and filter string in the fields down at the bottom of the dialog box).

Filter name: You can change the name of the currently selected filter here.

Note!

The filter name will only be used in this dialog to identify the filter for your convenience, it will not be used elsewhere. You can add multiple filters with the same name, but this is not very useful.

Filter string:	You can change the filter string of the currently selected filter here. Display Filter only: the string will be syntax checked while you are typing.
Add Expression...	Display Filter only: This button brings up the Add Expression dialog box which assists in building filter strings. You can find more information about the Add Expression dialog in Section 6.5, “The "Filter Expression" dialog box”
OK	Display Filter only: This button applies the selected filter to the current display and closes the dialog.
Apply	Display Filter only: This button applies the selected filter to the current display, and keeps the dialog open.
Save	Save the current settings in this dialog. The file location and format is explained in Appendix A, *Files and Folders*.
Close	Close this dialog. This will discard unsaved settings.

6.7. Defining and saving filter macros

You can define filter macros with Wireshark and give them labels for later use. This can save time in remembering and retyping some of the more complex filters you use.

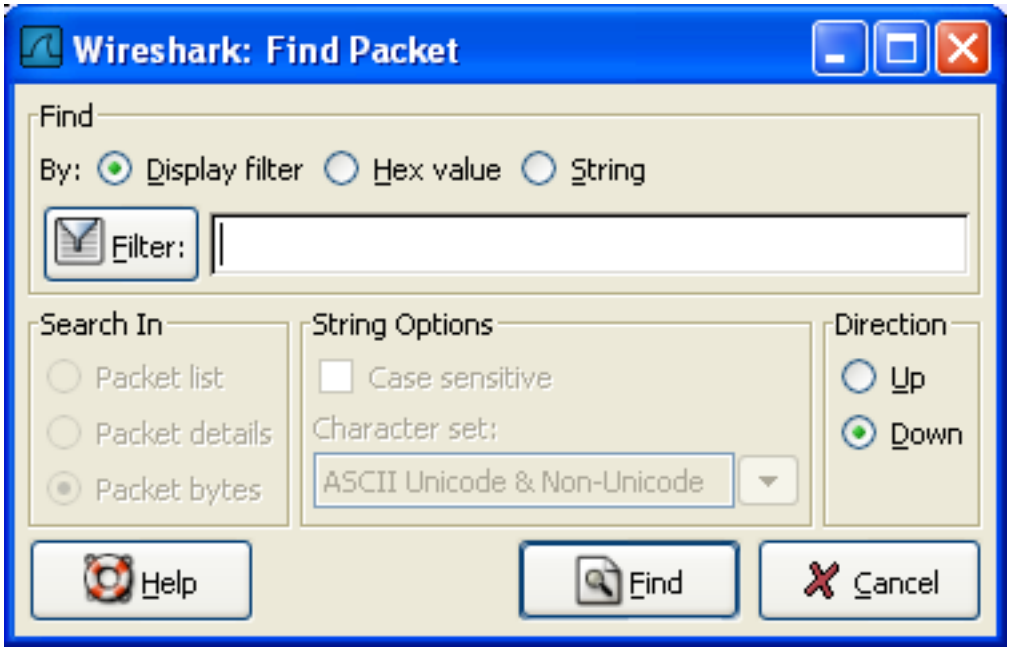
XXX - add an explanation of this.

6.8. Finding packets

You can easily find packets once you have captured some packets or have read in a previously saved capture file. Simply select the **Find Packet...** menu item from the **Edit** menu. Wireshark will pop up the dialog box shown in Figure 6.8, “The "Find Packet" dialog box”.

6.8.1. The "Find Packet" dialog box

Figure 6.8. The "Find Packet" dialog box



You might first select the kind of thing to search for:

- **Display filter**

  Simply enter a display filter string into the **Filter:** field, select a direction, and click on OK.

  For example, to find the three way handshake for a connection from host 192.168.0.1, use the following filter string:

  ```
  ip.src==192.168.0.1 and tcp.flags.syn==1
  ```

  For more details on display filters, see Section 6.3, “Filtering packets while viewing”

- **Hex Value**

  Search for a specific byte sequence in the packet data.

  For example, use "00:00" to find the next packet including two null bytes in the packet data.

- **String**

  Find a string in the packet data, with various options.

The value to be found will be syntax checked while you type it in. If the syntax check of your value succeeds, the background of the entry field will turn green, if it fails, it will turn red.

You can choose the search direction:

- **Up**

  Search upwards in the packet list (decreasing packet numbers).

- **Down**

  Search downwards in the packet list (increasing packet numbers).

6.8.2. The "Find Next" command

"Find Next" will continue searching with the same options used in the last "Find Packet".

6.8.3. The "Find Previous" command

"Find Previous" will do the same thing as "Find Next", but with reverse search direction.

6.9. Go to a specific packet

You can easily jump to specific packets with one of the menu items in the Go menu.

6.9.1. The "Go Back" command

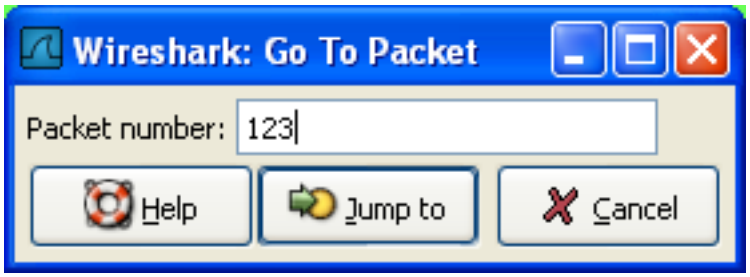
Go back in the packet history, works much like the page history in current web browsers.

6.9.2. The "Go Forward" command

Go forward in the packet history, works much like the page history in current web browsers.

6.9.3. The "Go to Packet" dialog box

Figure 6.9. The "Go To Packet" dialog box



This dialog box will let you enter a packet number. When you press **OK**, Wireshark will jump to that packet.

6.9.4. The "Go to Corresponding Packet" command

If a protocol field is selected which points to another packet in the capture file, this command will jump to that packet.

Note!

As these protocol fields now work like links (just as in your Web browser), it's easier to simply double-click on the field to jump to the corresponding field.

6.9.5. The "Go to First Packet" command

This command will simply jump to the first packet displayed.

6.9.6. The "Go to Last Packet" command

This command will simply jump to the last packet displayed.

6.10. Marking packets

You can mark packets in the "Packet List" pane. A marked packet will be shown with black background, regardless of the coloring rules set. Marking a packet can be useful to find it later while analyzing in a large capture file.

Warning!

The packet marks are not stored in the capture file or anywhere else, so all packet marks will be lost if you close the capture file.

You can use packet marking to control the output of packets when saving/exporting/printing. To do so, an option in the packet range is available, see Section 5.8, “The Packet Range frame”.

There are three functions to manipulate the marked state of a packet:

- **Mark packet (toggle)** toggles the marked state of a single packet.
- **Mark all packets** set the mark state of all packets.
- **Unmark all packets** reset the mark state of all packets.

These mark function are available from the "Edit" menu, and the "Mark packet (toggle)" function is also available from the pop-up menu of the "Packet List" pane.

6.11. Time display formats and time references

While packets are captured, each packet is timestamped. These timestamps will be saved to the capture file, so they will be available for later analysis.

A detailed description of timestamps, timezones and alike can be found at: Section 7.4, “Time Stamps”.

The timestamp presentation format and the precision in the packet list can be chosen using the View menu, see Figure 3.5, “The "View" Menu”.

The available presentation formats are:

- **Date and Time of Day: 1970-01-01 01:02:03.123456** The absolute date and time of the day when the packet was captured.
- **Time of Day: 01:02:03.123456** The absolute time of the day when the packet was captured.
- **Seconds Since Beginning of Capture: 123.123456** The time relative to the start of the capture file or the first "Time Reference" before this packet (see Section 6.11.1, “Packet time referencing”).
- **Seconds Since Previous Captured Packet: 1.123456** The time relative to the previous captured packet.
- **Seconds Since Previous Displayed Packet: 1.123456** The time relative to the previous displayed packet.
- **Seconds Since Epoch (1970-01-01): 1234567890.123456** The time relative to epoch (midnight UTC of January 1, 1970).

The available precisions (aka. the number of displayed decimal places) are:

- **Automatic** The timestamp precision of the loaded capture file format will be used (the default).
- **Seconds, Deciseconds, Centiseconds, Milliseconds, Microseconds or Nanoseconds** The timestamp precision will be forced to the given setting. If the actually available precision is smaller, zeros will be appended. If the precision is larger, the remaining decimal places will be cut off.

Precision example: If you have a timestamp and it's displayed using, "Seconds Since Previous Packet", : the value might be 1.123456. This will be displayed using the "Automatic" setting for libpcap files (which is microseconds). If you use Seconds it would show simply 1 and if you use Nanoseconds it shows 1.123456000.

6.11.1. Packet time referencing

The user can set time references to packets. A time reference is the starting point for all subsequent packet time calculations. It will be useful, if you want to see the time values relative to a special packet, e.g. the start of a new request. It's possible to set multiple time references in the capture file.

Warning!

The time references will not be saved permanently and will be lost when you close the capture file.

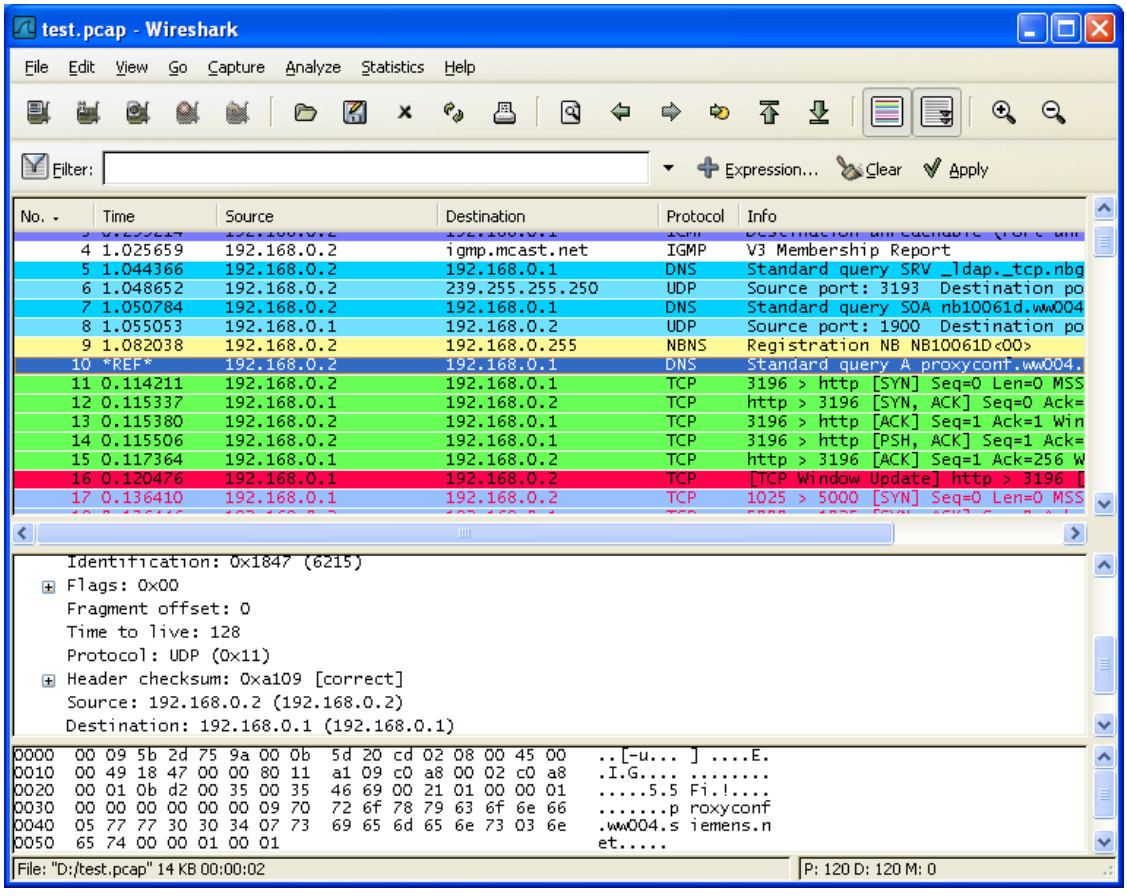
Note!

Time referencing will only be useful, if the time display format is set to "Seconds Since Beginning of Capture". If one of the other time display formats are used, time referencing will have no effect (and will make no sense either).

To work with time references, choose one of the "Time Reference" items in the "Edit" menu , see Section 3.6, “The "Edit" menu”, or from the pop-up menu of the "Packet List" pane.

- **Set Time Reference (toggle)** Toggles the time reference state of the currently selected packet to on or off.
- **Find Next** Find the next time referenced packet in the "Packet List" pane.
- **Find Previous** Find the previous time referenced packet in the "Packet List" pane.

Figure 6.10. Wireshark showing a time referenced packet



A time referenced packet will be marked with the string *REF* in the Time column (see packet number 10). All subsequent packets will show the time since the last time reference.

Chapter 7. Advanced Topics

7.1. Introduction

In this chapter some of the advanced features of Wireshark will be described.

7.2. Following TCP streams

If you are working with TCP based protocols it can be very helpful to see the data from a TCP stream in the way that the application layer sees it. Perhaps you are looking for passwords in a Telnet stream, or you are trying to make sense of a data stream. Maybe you just need a display filter to show only the packets of that TCP stream. If so, Wireshark's ability to follow a TCP stream will be useful to you.

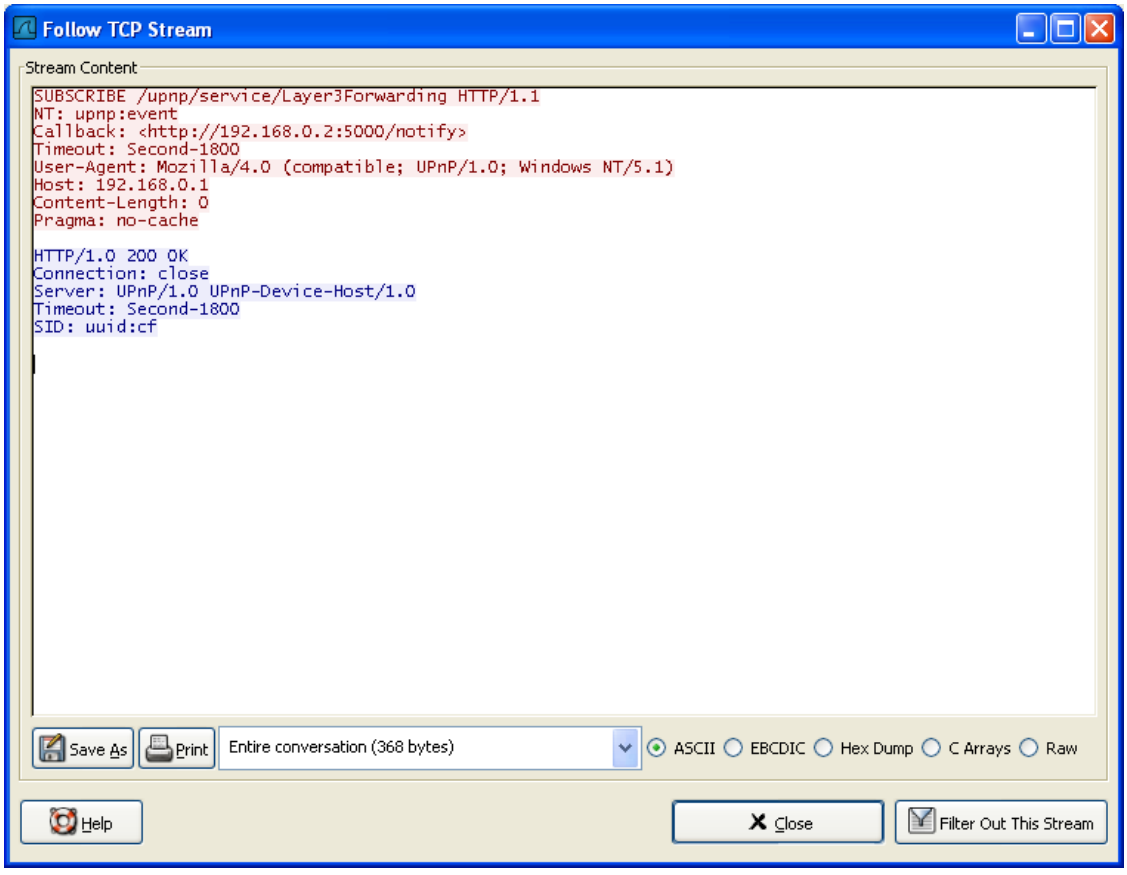
Simply select a TCP packet in the packet list of the stream/connection you are interested in and then select the Follow TCP Stream menu item from the Wireshark Tools menu (or use the context menu in the packet list). Wireshark will set an appropriate display filter and pop up a dialog box with all the data from the TCP stream laid out in order, as shown in Figure 7.1, “The "Follow TCP Stream" dialog box”.

Note!

It is worthwhile noting that Follow TCP Stream installs a display filter to select all the packets in the TCP stream you have selected.

7.2.1. The "Follow TCP Stream" dialog box

Figure 7.1. The "Follow TCP Stream" dialog box



The stream content is displayed in the same sequence as it appeared on the network. Traffic from A to B

is marked in red, while traffic from B to A is marked in blue. If you like, you can change these colors in the Edit/Preferences "Colors" page.

Non-printable characters will be replaced by dots. XXX - What about line wrapping (maximum line length) and CRNL conversions?

The stream content won't be updated while doing a live capture. To get the latest content you'll have to reopen the dialog.

You can choose from the following actions:

1. **Save As**: Save the stream data in the currently selected format.
2. **Print**: Print the stream data in the currently selected format.
3. **Direction**: Choose the stream direction to be displayed ("Entire conversation", "data from A to B only" or "data from B to A only").
4. **Filter out this stream**: Apply a display filter removing the current TCP stream data from the display.
5. **Close**: Close this dialog box, leaving the current display filter in effect.

You can choose to view the data in one of the following formats:

1. **ASCII**: In this view you see the data from each direction in ASCII. Obviously best for ASCII based protocols, e.g. HTTP.
2. **EBCDIC**: For the big-iron freaks out there.
3. **HEX Dump**: This allows you to see all the data. This will require a lot of screen space and is best used with binary protocols.
4. **C Arrays**: This allows you to import the stream data into your own C program.
5. **Raw**: This allows you to load the unaltered stream data into a different program for further examination. The display will look the same as the ASCII setting, but "Save As" will result in a binary file.

7.3. Expert Infos

The expert infos is a kind of log of the anomalies found by Wireshark in a capture file.

The general idea behind the following "Expert Info" is to have a better display of "uncommon" or just notable network behaviour. This way, both novice and expert users will hopefully find probable network problems a lot faster, compared to scanning the packet list "manually" .

Expert infos are only a hint!

Take expert infos as a hint what's worth looking at, but not more. For example: The absence of expert infos doesn't necessarily mean everything is ok!

The amount of expert infos largely depends on the protocol being used!

While some common protocols like TCP/IP will show detailed expert infos, most other protocols currently won't show any expert infos at all.

The following will first describe the components of a single expert info, then the User Interface.

7.3.1. Expert Info Entries

Each expert info will contain the following things which will be described in detail below:

Table 7.1. Some example expert infos

Packet #	Severity	Group	Protocol	Summary		
1	Note	Sequence	TCP	Duplicate ACK (#1)		
2	Chat	Sequence	TCP	Connection reset (RST)		
8	Note	Sequence	TCP	Keep-Alive		
9	Warn	Sequence	TCP	Fast retransmission (suspected)		

7.3.1.1. Severity

Every expert info has a specific severity level. The following severity levels are used, in parentheses are the colors in which the items will be marked in the GUI:

- **Chat (grey)**: information about usual workflow, e.g. a TCP packet with the SYN flag set
- **Note (cyan)**: notable things, e.g. an application returned an "usual" error code like HTTP 404
- **Warn (yellow)**: warning, e.g. application returned an "unusual" error code like a connection problem

- **Error (red)**: serious problem, e.g. [Malformed Packet]

7.3.1.2. Group

There are some common groups of expert infos. The following are currently implemented:

- **Checksum**: a checksum was invalid
- **Sequence**: protocol sequence suspicious, e.g. sequence wasn't continuous or a retransmission was detected or ...
- **Response Code**: problem with application response code, e.g. HTTP 404 page not found
- **Request Code**: an application request (e.g. File Handle == x), usually Chat level
- **Undecoded**: dissector incomplete or data can't be decoded for other reasons
- **Reassemble**: problems while reassembling, e.g. not all fragments were available or an exception happened while reassembling
- **Malformed**: malformed packet or dissector has a bug, dissection of this packet aborted
- **Debug**: debugging (should not occur in release versions)

It's possible that more such group values will be added in the future ...

7.3.1.3. Protocol

The protocol in which the expert info was caused.

7.3.1.4. Summary

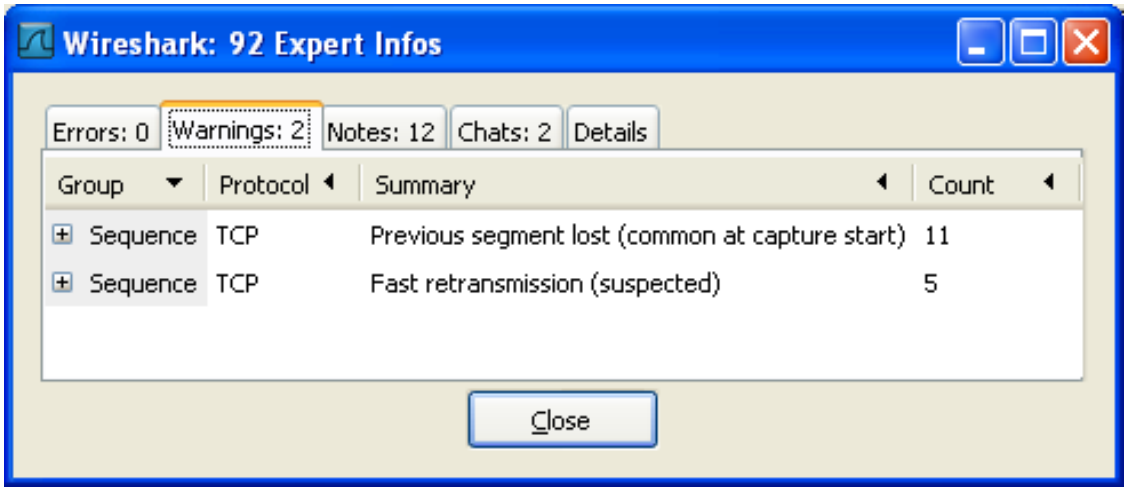
Each expert info will also have a short additional text with some further explanation.

7.3.2. "Expert Info Composite" dialog

From the main menu you can open the expert info dialog, using: "Analyze/Expert Info Composite"

XXX - "Analyze/Expert Info" also exists but is subject to removal and therefore not explained here.

XXX - add explanation of the dialogs context menu.



7.3.2.1. Errors / Warnings / Notes / Chats tabs

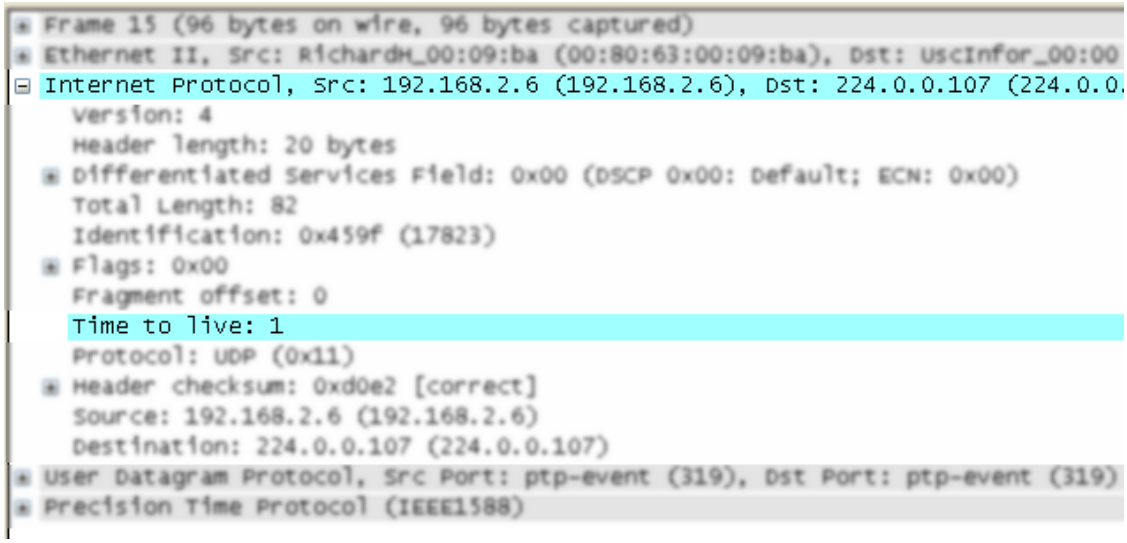
An easy and quick way to find the most interesting infos (rather than using the Details tab), is to have a look at the separate tabs for each severity level. As the tab label also contains the number of existing entries, it's easy to find the tab with the most important entries.

There are usually a lot of identical expert infos only differing in the packet number. These identical infos will be combined into a single line - with a count column showing how often they appeared in the capture file. Clicking on the plus sign shows the individual packet numbers in a tree view.

7.3.2.2. Details tab

The Details tab provides the expert infos in a "log like" view, each entry on its own line (much like the packet list). As the amount of expert infos for a capture file can easily become very large, getting an idea of the interesting infos with this view can take quite a while. The advantage of this tab is to have all entries in the sequence as they appeared, this is sometimes a help to pinpoint problems.

7.3.3. "Colorized" Protocol Details Tree

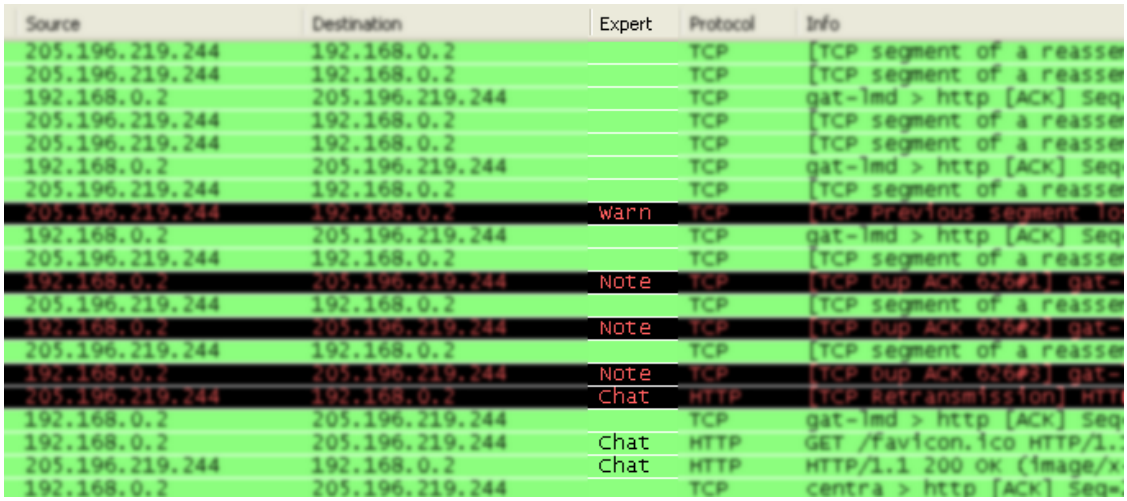

Frame 15 (96 bytes on wire, 96 bytes captured)
Ethernet II, Src: RichardH_00:09:ba (00:80:63:00:09:ba), Dst: UscInfor_00:00
Internet Protocol, Src: 192.168.2.6 (192.168.2.6), Dst: 224.0.0.107 (224.0.0.
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 82
Identification: 0x459f (17823)
Flags: 0x00
Fragment offset: 0
Time to live: 1
Protocol: UDP (0x11)
Header checksum: 0xd0e2 [correct]
Source: 192.168.2.6 (192.168.2.6)
Destination: 224.0.0.107 (224.0.0.107)
User Datagram Protocol, Src Port: ptp-event (319), Dst Port: ptp-event (319)
Precision Time Protocol (IEEE1588)

The protocol field causing an expert info is colorized, e.g. uses a cyan background for a note severity level. This color is propagated to the toplevel protocol item in the tree, so it's easy to find the field that

caused the expert info.

For the example screenshot above, the IP "Time to live" value is very low (only 1), so the corresponding protocol field is marked with a cyan background. To easier find that item in the packet tree, the IP protocol toplevel item is marked cyan as well.

7.3.4. "Expert" Packet List Column (optional)


Source
Destination
Expert
Protocol
Info
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
192.168.0.2 205.196.219.244 TCP gat-lmd > http [ACK] Seq
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
192.168.0.2 205.196.219.244 TCP gat-lmd > http [ACK] Seq
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
Warn TCP [TCP Previous segment lo
192.168.0.2 205.196.219.244 TCP gat-lmd > http [ACK] Seq
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
Note TCP [TCP Dup ACK 626#1] gat-
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
Note TCP [TCP Dup ACK 626#2] gat-
205.196.219.244 192.168.0.2 TCP [TCP segment of a reasse
Note TCP [TCP Dup ACK 626#3] gat-
Chat HTTP [TCP Retransmission] HTT
192.168.0.2 205.196.219.244 TCP gat-lmd > http [ACK] Seq
192.168.0.2 205.196.219.244 Chat HTTP GET /favicon.ico HTTP/1.
205.196.219.244 192.168.0.2 Chat HTTP HTTP/1.1 200 OK (image/x
192.168.0.2 205.196.219.244 TCP centra > http [ACK] Seq=

An optional "Expert Info Severity" packet list column is available (since SVN 22387 -> 0.99.7), that displays the most significant severity of a packet, or stays empty if everything seems ok. This column is not displayed by default, but can be easily added using the Preferences Columns page described in Section 9.5, “Preferences”.

7.4. Time Stamps

Time stamps, their precisions and all that can be quite confusing. This section will provide you with information about what's going on while Wireshark processes time stamps.

While packets are captured, each packet is time stamped as it comes in. These time stamps will be saved to the capture file, so they also will be available for (later) analysis.

So where do these time stamps come from? While capturing, Wireshark gets the time stamps from the libpcap (WinPcap) library, which in turn gets them from the operating system kernel. If the capture data is loaded from a capture file, Wireshark obviously gets the data from that file.

7.4.1. Wireshark internals

The internal format that Wireshark uses to keep a packet time stamp consists of the date (in days since 1.1.1970) and the time of day (in nanoseconds since midnight). You can adjust the way Wireshark displays the time stamp data in the packet list, see the "Time Display Format" item in the Section 3.7, “The "View" menu” for details.

While reading or writing capture files, Wireshark converts the time stamp data between the capture file format and the internal format as required.

While capturing, Wireshark uses the libpcap (WinPcap) capture library which supports microsecond resolution. Unless you are working with specialized capturing hardware, this resolution should be adequate.

7.4.2. Capture file formats

Every capture file format that Wireshark knows supports time stamps. The time stamp precision supported by a specific capture file format differs widely and varies from one second "0" to one nanosecond "0.123456789". Most file formats store the time stamps with a fixed precision (e.g. microseconds), while some file formats are even capable of storing the time stamp precision itself (whatever the benefit may be).

The common libpcap capture file format that is used by Wireshark (and a lot of other tools) supports a fixed microsecond resolution "0.123456" only.

Note!

Writing data into a capture file format that doesn't provide the capability to store the actual precision will lead to loss of information. Example: If you load a capture file with nanosecond resolution and store the capture data to a libpcap file (with microsecond resolution) Wireshark obviously must reduce the precision from nanosecond to microsecond.

7.4.3. Accuracy

It's often asked: "Which time stamp accuracy is provided by Wireshark?". Well, Wireshark doesn't create any time stamps itself but simply gets them from "somewhere else" and displays them. So accuracy will depend on the capture system (operating system, performance, ...) that you use. Because of this, the above question is difficult to answer in a general way.

Note!

USB connected network adapters often provide a very bad time stamp accuracy. The incoming packets have to take "a long and winding road" to travel through the USB cable

until they actually reach the kernel. As the incoming packets are time stamped when they are processed by the kernel, this time stamping mechanism becomes very inaccurate.

Conclusion: don't use USB connected NIC's when you need precise time stamp accuracy! (XXX - are there any such NIC's that generate time stamps on the USB hardware?)

7.5. Time Zones

If you travel across the planet, time zones can be confusing. If you get a capture file from somewhere around the world time zones can even be a lot more confusing ;-)

First of all, there are two reasons why you may not need to think about time zones at all:

- You are only interested in the time differences between the packet time stamps and don't need to know the exact date and time of the captured packets (which is often the case).
- You don't get capture files from different time zones than your own, so there are simply no time zone problems. For example: everyone in your team is working in the same time zone as yourself.

What are time zones?

People expect that the time reflects the sunset. Dawn should be in the morning maybe around 06:00 and dusk in the evening maybe at 20:00. These times will obviously vary depending on the season. It would be very confusing if everyone on earth would use the same global time as this would correspond to the sunset only at a small part of the world.

For that reason, the earth is split into several different time zones, each zone with a local time that corresponds to the local sunset.

The time zone's base time is UTC (Coordinated Universal Time) or Zulu Time (military and aviation). The older term GMT (Greenwich Mean Time) shouldn't be used as it is slightly incorrect (up to 0.9 seconds difference to UTC). The UTC base time equals to 0 (based at Greenwich, England) and all time zones have an offset to UTC between -12 to +14 hours!

For example: If you live in Berlin you are in a time zone one hour earlier than UTC, so you are in time zone "+1" (time difference in hours compared to UTC). If it's 3 o'clock in Berlin it's 2 o'clock in UTC "at the same moment".

Be aware that at a few places on earth don't use time zones with even hour offsets (e.g. New Delhi uses UTC+05:30)!

Further information can be found at: http://en.wikipedia.org/wiki/Time_zone and http://en.wikipedia.org/wiki/Coordinated_Universal_Time.

What is daylight saving time (DST)?

Daylight Saving Time (DST), also known as Summer Time, is intended to "save" some daylight during the summer months. To do this, a lot of countries (but not all!) add a DST hour to the already existing UTC offset. So you may need to take another hour (or in very rare cases even two hours!) difference into your "time zone calculations".

Unfortunately, the date at which DST actually takes effect is different throughout the world. You may also note, that the northern and southern hemispheres have opposite DST's (e.g. while it's summer in Europe it's winter in Australia).

Keep in mind: UTC remains the same all year around, regardless of DST!

Further information can be found at: http://en.wikipedia.org/wiki/Daylight_saving.

Further time zone and DST information can be found at: http://wwp.greenwichmeantime.com/ and http://www.timeanddate.com/worldclock/.

7.5.1. Set your computer's time correctly!

If you work with people around the world, it's very helpful to set your computer's time and time zone right.

You should set your computers time and time zone in the correct sequence:

1. Set your time zone to your current location
2. Set your computer's clock to the local time

This way you will tell your computer both the local time and also the time offset to UTC.

Tip!

If you travel around the world, it's an often made mistake to adjust the hours of your computer clock to the local time. Don't adjust the hours but your time zone setting instead! For your computer, the time is essentially the same as before, you are simply in a different time zone with a different local time!

Tip!

You can use the Network Time Protocol (NTP) to automatically adjust your computer to the correct time, by synchronizing it to Internet NTP clock servers. NTP clients are available for all operating systems that Wireshark supports (and for a lot more), for examples see: http://www.ntp.org/.

7.5.2. Wireshark and Time Zones

So what's the relationship between Wireshark and time zones anyway?

Wireshark's native capture file format (libpcap format), and some other capture file formats, such as the Windows Sniffer, EtherPeek, AiroPeek, and Sun snoop formats, save the arrival time of packets as UTC values. UN*X systems, and "Windows NT based" systems (Windows NT 4.0, 2000, XP, Server 2003, Vista, Server 2008) represent time internally as UTC. When Wireshark is capturing, no conversion is necessary. However, if the system time zone is not set correctly, the system's UTC time might not be correctly set even if the system clock appears to display correct local time. "Windows 9x based" systems (Windows 95, Windows 98, Windows Me) represent time internally as local time. When capturing, WinPcap has to convert the time to UTC before supplying it to Wireshark. If the system's time zone is not set correctly, that conversion will not be done correctly.

Other capture file formats, such as the Microsoft Network Monitor, DOS-based Sniffer, and Network Instruments Observer formats, save the arrival time of packets as local time values.

Internally to Wireshark, time stamps are represented in UTC; this means that, when reading capture files that save the arrival time of packets as local time values, Wireshark must convert those local time values to UTC values.

Wireshark in turn will display the time stamps always in local time. The displaying computer will convert them from UTC to local time and displays this (local) time. For capture files saving the arrival time of packets as UTC values, this means that the arrival time will be displayed as the local time in your

time zone, which might not be the same as the arrival time in the time zone in which the packet was captured. For capture files saving the arrival time of packets as local time values, the conversion to UTC will be done using your time zone's offset from UTC and DST rules, which means the conversion will not be done correctly; the conversion back to local time for display might undo this correctly, in which case the arrival time will be displayed as the arrival time in which the packet was captured.

Table 7.2. Time zone examples for UTC arrival times (without DST)

	Los Angeles	New York	Madrid	London	Berlin	Tokyo
Capture File (UTC)	10:00	10:00	10:00	10:00	10:00	10:00
Local Offset to UTC	-8	-5	-1	0	+1	+9
Displayed Time (Local Time)	02:00	05:00	09:00	10:00	11:00	19:00

An example: Let's assume that someone in Los Angeles captured a packet with Wireshark at exactly 2 o'clock local time and sends you this capture file. The capture file's time stamp will be represented in UTC as 10 o'clock. You are located in Berlin and will see 11 o'clock on your Wireshark display.

Now you have a phone call, video conference or Internet meeting with that one to talk about that capture file. As you are both looking at the displayed time on your local computers, the one in Los Angeles still sees 2 o'clock but you in Berlin will see 11 o'clock. The time displays are different as both Wireshark displays will show the (different) local times at the same point in time.

Conclusion: You may not bother about the date/time of the time stamp you currently look at, unless you must make sure that the date/time is as expected. So, if you get a capture file from a different time zone and/or DST, you'll have to find out the time zone/DST difference between the two local times and "mentally adjust" the time stamps accordingly. In any case, make sure that every computer in question has the correct time and time zone setting.

7.6. Packet Reassembling

7.6.1. What is it?

Network protocols often need to transport large chunks of data, which are complete in themselves, e.g. when transferring a file. The underlying protocol might not be able to handle that chunk size (e.g. limitation of the network packet size), or is stream-based like TCP, which doesn't know data chunks at all.

In that case the network protocol has to handle the chunk boundaries itself and (if required) spread the data over multiple packets. It obviously also needs a mechanism to determine the chunk boundaries on the receiving side.

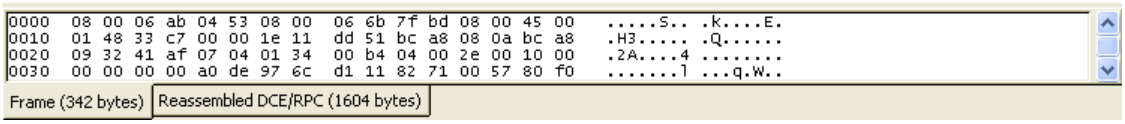
Tip!

Wireshark calls this mechanism reassembling, although a specific protocol specification might use a different term for this (e.g. desegmentation, defragmentation, ...).

7.6.2. How Wireshark handles it

For some of the network protocols Wireshark knows of, a mechanism is implemented to find, decode and display these chunks of data. Wireshark will try to find the corresponding packets of this chunk, and will show the combined data as additional pages in the "Packet Bytes" pane (for information about this pane, see Section 3.18, “The "Packet Bytes" pane”).

Figure 7.2. The "Packet Bytes" pane with a reassembled tab



Note!

Reassembling might take place at several protocol layers, so it's possible that multiple tabs in the "Packet Bytes" pane appear.

Note!

You will find the reassembled data in the last packet of the chunk.

An example: In a **HTTP** GET response, the requested data (e.g. an HTML page) is returned. Wireshark will show the hex dump of the data in a new tab "Uncompressed entity body" in the "Packet Bytes" pane.

Reassembling is enabled in the preferences by default. The defaults were changed from disabled to enabled in September 2005. If you created your preference settings before this date, you might look if reassembling is actually enabled, as it can be extremely helpful while analyzing network packets.

The enabling or disabling of the reassemble settings of a protocol typically requires two things:

1. the lower level protocol (e.g., TCP) must support reassembly. Often this reassembly can be enabled or disabled via the protocol preferences.

2. the higher level protocol (e.g., HTTP) must use the reassembly mechanism to reassemble fragmented protocol data. This too can often be enabled or disabled via the protocol preferences.

The tooltip of the higher level protocol setting will notify you if and which lower level protocol setting also has to be considered.

7.7. Name Resolution

Name resolution tries to resolve some of the numerical address values into a human readable format. There are two possible ways to do these conversations, depending on the resolution to be done: calling system/network services (like the gethostname function) and/or evaluate from Wireshark specific configuration files. For details about the configuration files Wireshark uses for name resolution and alike, see Appendix A, *Files and Folders*.

The name resolution feature can be en-/disabled separately for the protocol layers of the following sections.

7.7.1. Name Resolution drawbacks

Name resolution can be invaluable while working with Wireshark and may even save you hours of work. Unfortunately, it also has its drawbacks.

- **Name resolution will often fail.** The name to be resolved might simply be unknown by the name servers asked or the servers are just not available and the name is also not found in Wireshark's configuration files.
- **The resolved names are not stored in the capture file or somewhere else.** So the resolved names might not be available if you open the capture file later or on a different machine. Each time you open a capture file it may look "slightly different", maybe simply because you can't connect to a name server (which you could connect before).
- **DNS may add additional packets to your capture file.** You may see packets to/from your machine in your capture file, which are caused by name resolution network services of the machine Wireshark captures from. XXX - are there any other such packets than DNS ones?
- **Resolved DNS names are cached by Wireshark.** This is required for acceptable performance. However, if the name resolution information should change while Wireshark is running, Wireshark won't notice a change to the name resolution information once it gets cached. If this information changes while Wireshark is running, e.g. a new DHCP lease takes effect, Wireshark won't notice it. XXX - is this true for all or only for DNS info?

Tip!

The name resolution in the packet list is done while the list is filled. If a name could be resolved after a packet was added to the list, that former entry won't be changed. As the name resolution results are cached, you can use "View/Reload" to rebuild the packet list, this time with the correctly resolved names. However, this isn't possible while a capture is in progress.

7.7.2. Ethernet name resolution (MAC layer)

Try to resolve an Ethernet MAC address (e.g. 00:09:5b:01:02:03) to something more "human readable".

ARP name resolution (system service): Wireshark will ask the operating system to convert an Ethernet address to the corresponding IP address (e.g. 00:09:5b:01:02:03 -> 192.168.0.1).

Ethernet codes (ethers file): If the ARP name resolution failed, Wireshark tries to convert the Ethernet address to a known device name, which has been assigned by the user using an ethers file (e.g. 00:09:5b:01:02:03 -> homerouter).

Ethernet manufacturer codes (manuf file): If neither ARP or ethers returns a result, Wireshark tries to convert the first 3 bytes of an ethernet address to an abbreviated manufacturer name, which has been assigned by the IEEE (e.g. 00:09:5b:01:02:03 -> Netgear_01:02:03).

7.7.3. IP name resolution (network layer)

Try to resolve an IP address (e.g. 216.239.37.99) to something more "human readable".

DNS/ADNS name resolution (system/library service): Wireshark will ask the operating system (or the ADNS library), to convert an IP address to the hostname associated with it (e.g. 216.239.37.99 -> www.1.google.com). The DNS service is using synchronous calls to the DNS server. So Wireshark will stop responding until a response to a DNS request is returned. If possible, you might consider using the ADNS library (which won't wait for a network response).

Warning!

Enabling network name resolution when your name server is unavailable may significantly slow down Wireshark while it waits for all of the name server requests to time out. Use ADNS in that case.

DNS vs. ADNS: here's a short comparison: Both mechanisms are used to convert an IP address to some human readable (domain) name. The usual DNS call gethostname() will try to convert the address to a name. To do this, it will first ask the systems hosts file (e.g. /etc/hosts) if it finds a matching entry. If that fails, it will ask the configured DNS server(s) about the name.

So the real difference between DNS and ADNS comes when the system has to wait for the DNS server about a name resolution. The system call gethostname() will wait until a name is resolved or an error occurs. If the DNS server is unavailable, this might take quite a while (several seconds). The ADNS service will work a bit differently. It will also ask the DNS server, but it won't wait for the answer. It will just return to Wireshark in a very short amount of time. The actual (and the following) address fields won't show the resolved name until the ADNS call returned. As mentioned above, the values get cached, so you can use View/Reload to "update" these fields to show the resolved values.

hosts name resolution (hosts file): If DNS name resolution failed, Wireshark will try to convert an IP address to the hostname associated with it, using a hosts file provided by the user (e.g. 216.239.37.99 -> www.google.com).

7.7.4. IPX name resolution (network layer)

ipxnet name resolution (ipxnets file): XXX - add ipxnets name resolution explanation.

7.7.5. TCP/UDP port name resolution (transport layer)

Try to resolve a TCP/UDP port (e.g. 80) to something more "human readable".

TCP/UDP port conversion (system service): Wireshark will ask the operating system to convert a TCP or UDP port to its well known name (e.g. 80 -> http).

XXX - mention the role of the /etc/services file (but don't forget the files and folders section)!

7.8. Checksums

Several network protocols use checksums to ensure data integrity.

Tip!

Applying checksums as described here is also known as **redundancy checking**.

What are checksums for?

Checksums are used to ensure the integrity of data portions for data transmission or storage. A checksum is basically a calculated summary of such a data portion.

Network data transmissions often produce errors, such as toggled, missing or duplicated bits. As a result, the data received might not be identical to the data transmitted, which is obviously a bad thing.

Because of these transmission errors, network protocols very often use checksums to detect such errors. The transmitter will calculate a checksum of the data and transmits the data together with the checksum. The receiver will calculate the checksum of the received data with the same algorithm as the transmitter. If the received and calculated checksums don't match a transmission error has occurred.

Some checksum algorithms are able to recover (simple) errors by calculating where the expected error must be and repairing it.

If there are errors that cannot be recovered, the receiving side throws away the packet. Depending on the network protocol, this data loss is simply ignored or the sending side needs to detect this loss somehow and retransmits the required packet(s).

Using a checksum drastically reduces the number of undetected transmission errors. However, the usual checksum algorithms cannot guarantee an error detection of 100%, so a very small number of transmission errors may remain undetected.

There are several different kinds of checksum algorithms; an example of an often used checksum algorithm is CRC32. The checksum algorithm actually chosen for a specific network protocol will depend on the expected error rate of the network medium, the importance of error detection, the processor load to perform the calculation, the performance needed and many other things.

Further information about checksums can be found at: http://en.wikipedia.org/wiki/Checksum.

7.8.1. Wireshark checksum validation

Wireshark will validate the checksums of several protocols, e.g.: IP, TCP, UDP, ...

It will do the same calculation as a "normal receiver" would do, and shows the checksum fields in the packet details with a comment, e.g.: [correct], [invalid, must be 0x12345678] or alike.

Checksum validation can be switched off for various protocols in the Wireshark protocol preferences, e.g. to (very slightly) increase performance.

If the checksum validation is enabled and it detected an invalid checksum, features like packet reassembling won't be processed. This is avoided as incorrect connection data could "confuse" the internal database.

7.8.2. Checksum offloading

The checksum calculation might be done by the network driver, protocol driver or even in hardware.

For example: The Ethernet transmitting hardware calculates the Ethernet CRC32 checksum and the receiving hardware validates this checksum. If the received checksum is wrong Wireshark won't even see the packet, as the Ethernet hardware internally throws away the packet.

Higher level checksums are "traditionally" calculated by the protocol implementation and the completed packet is then handed over to the hardware.

Recent network hardware can perform advanced features such as IP checksum calculation, also known as checksum offloading. The network driver won't calculate the checksum itself but will simply hand over an empty (zero or garbage filled) checksum field to the hardware.

Note!

Checksum offloading often causes confusion as the network packets to be transmitted are handed over to Wireshark before the checksums are actually calculated. Wireshark gets these "empty" checksums and displays them as invalid, even though the packets will contain valid checksums when they leave the network hardware later.

Checksum offloading can be confusing and having a lot of [invalid] messages on the screen can be quite annoying. As mentioned above, invalid checksums may lead to unreassembled packets, making the analysis of the packet data much harder.

You can do two things to avoid this checksum offloading problem:

- Turn off the checksum offloading in the network driver, if this option is available.
- Turn off checksum validation of the specific protocol in the Wireshark preferences.

Chapter 8. Statistics

8.1. Introduction

Wireshark provides a wide range of network statistics which can be accessed via the **Statistics** menu.

These statistics range from general information about the loaded capture file (like the number of captured packets), to statistics about specific protocols (e.g. statistics about the number of HTTP requests and responses captured).

- General statistics:
  - **Summary** about the capture file.
  - **Protocol Hierarchy** of the captured packets.
  - **Conversations** e.g. traffic between specific IP addresses.
  - **Endpoints** e.g. traffic to and from an IP addresses.
  - **IO Graphs** visualizing the number of packets (or similar) in time.
- Protocol specific statistics:
  - **Service Response Time** between request and response of some protocols.
  - **Various other** protocol specific statistics.

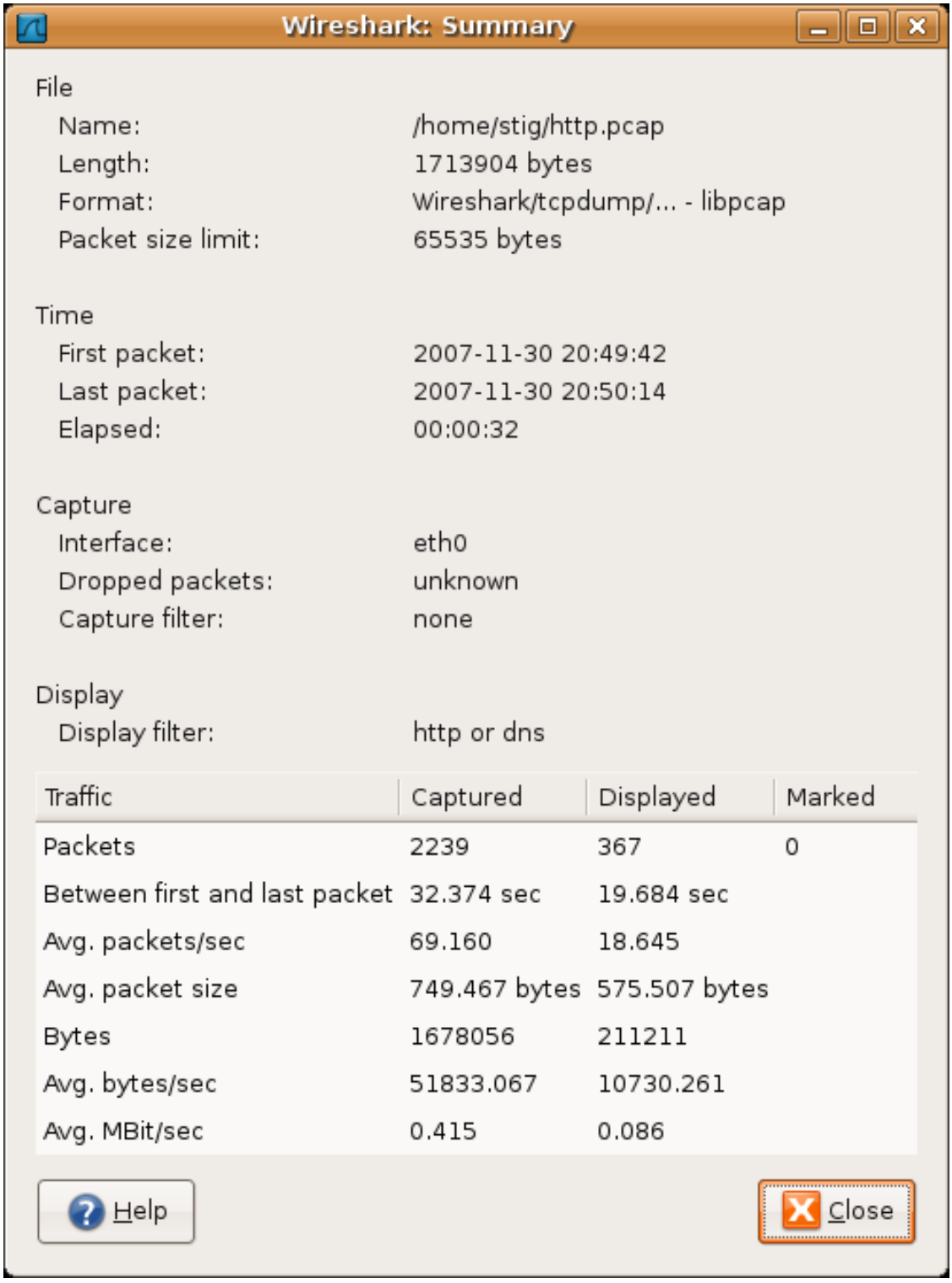
Note!

The protocol specific statistics requires detailed knowledge about the specific protocol. Unless you are familiar with that protocol, statistics about it will be pretty hard to understand.

8.2. The "Summary" window

General statistics about the current capture file.

Figure 8.1. The "Summary" window



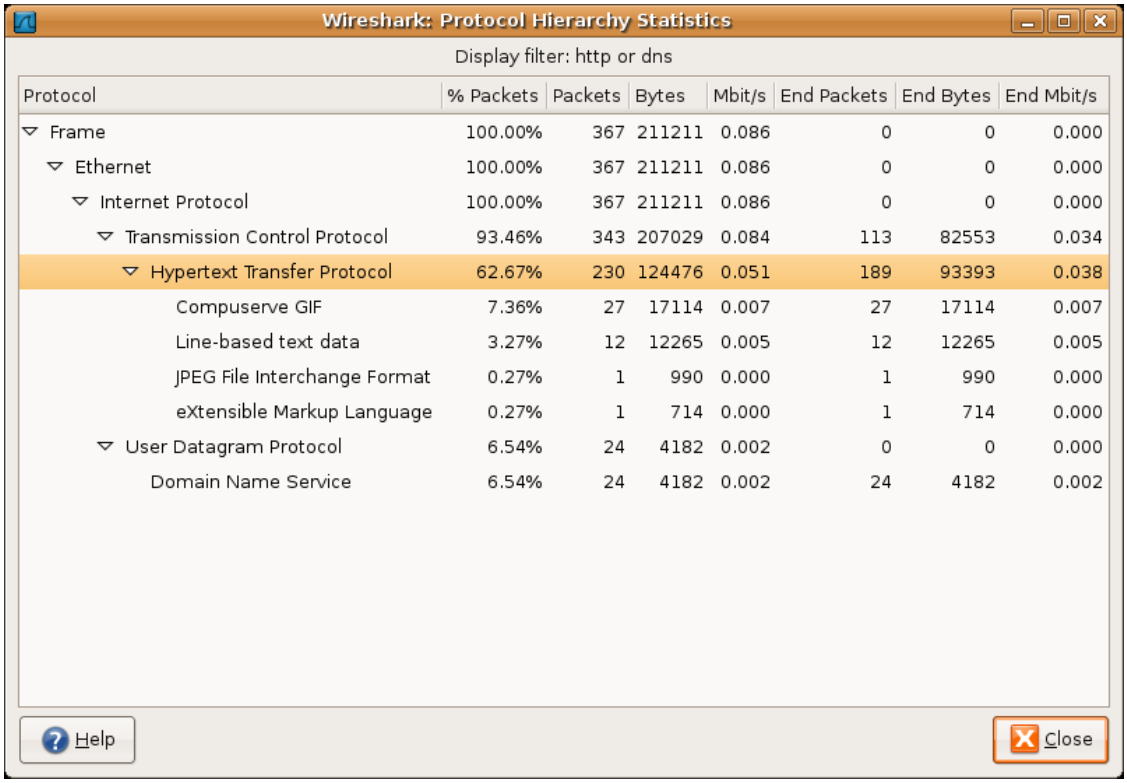
- **File**: general information about the capture file.
- **Time**: the timestamps when the first and the last packet were captured (and the time between them).

- **Capture**: information from the time when the capture was done (only available if the packet data was captured from the network and not loaded from a file).
- **Display**: some display related information.
- **Traffic**: some statistics of the network traffic seen. If a display filter is set, you will see values in the Captured column, and if any packages are marked, you will see values in the Marked column. The values in the **Captured** column will remain the same as before, while the values in the **Displayed** column will reflect the values corresponding to the packets shown in the display. The values in the **Marked** column will reflect the values corresponding to the marked packages.

8.3. The "Protocol Hierarchy" window

The protocol hierarchy of the captured packets.

Figure 8.2. The "Protocol Hierarchy" window


Wireshark: Protocol Hierarchy Statistics
Display filter: http or dns
Protocol
% Packets
Packets
Bytes
Mbit/s
End Packets
End Bytes
End Mbit/s
Frame 100.00% 367 211211 0.086 0 0 0.000
Ethernet 100.00% 367 211211 0.086 0 0 0.000
Internet Protocol 100.00% 367 211211 0.086 0 0 0.000
Transmission Control Protocol 93.46% 343 207029 0.084 113 82553 0.034
Hypertext Transfer Protocol 62.67% 230 124476 0.051 189 93393 0.038
Compuserve GIF 7.36% 27 17114 0.007 27 17114 0.007
Line-based text data 3.27% 12 12265 0.005 12 12265 0.005
JPEG File Interchange Format 0.27% 1 990 0.000 1 990 0.000
eXtensible Markup Language 0.27% 1 714 0.000 1 714 0.000
User Datagram Protocol 6.54% 24 4182 0.002 0 0 0.000
Domain Name Service 6.54% 24 4182 0.002 24 4182 0.002
Help
Close

This is a tree of all the protocols in the capture. You can collapse or expand subtrees, by clicking on the plus / minus icons. By default, all trees are expanded.

Each row contains the statistical values of one protocol. The **Display filter** will show the current display filter.

The following columns containing the statistical values are available:

- **Protocol**: this protocol's name
- **% Packets**: the percentage of protocol packets, relative to all packets in the capture
- **Packets**: the absolute number of packets of this protocol
- **Bytes**: the absolute number of bytes of this protocol
- **MBit/s**: the bandwidth of this protocol, relative to the capture time
- **End Packets**: the absolute number of packets of this protocol (where this protocol was the highest protocol to decode)
- **End Bytes**: the absolute number of bytes of this protocol (where this protocol was the highest pro-

tocol to decode)

- **End MBit/s**: the bandwidth of this protocol, relative to the capture time (where this protocol was the highest protocol to decode)

Note!

Packets will usually contain multiple protocols, so more than one protocol will be counted for each packet. Example: In the screenshot IP has 99,17% and TCP 85,83% (which is together much more than 100%).

Note!

Protocol layers can consist of packets that won't contain any higher layer protocol, so the sum of all higher layer packets may not sum up to the protocols packet count. Example: In the screenshot TCP has 85,83% but the sum of the subprotocols (HTTP, ...) is much less. This may be caused by TCP protocol overhead, e.g. TCP ACK packets won't be counted as packets of the higher layer).

Note!

A single packet can contain the same protocol more than once. In this case, the protocol is counted more than once. For example: in some tunneling configurations the IP layer can appear twice.

8.4. Conversations

Statistics of the captured conversations.

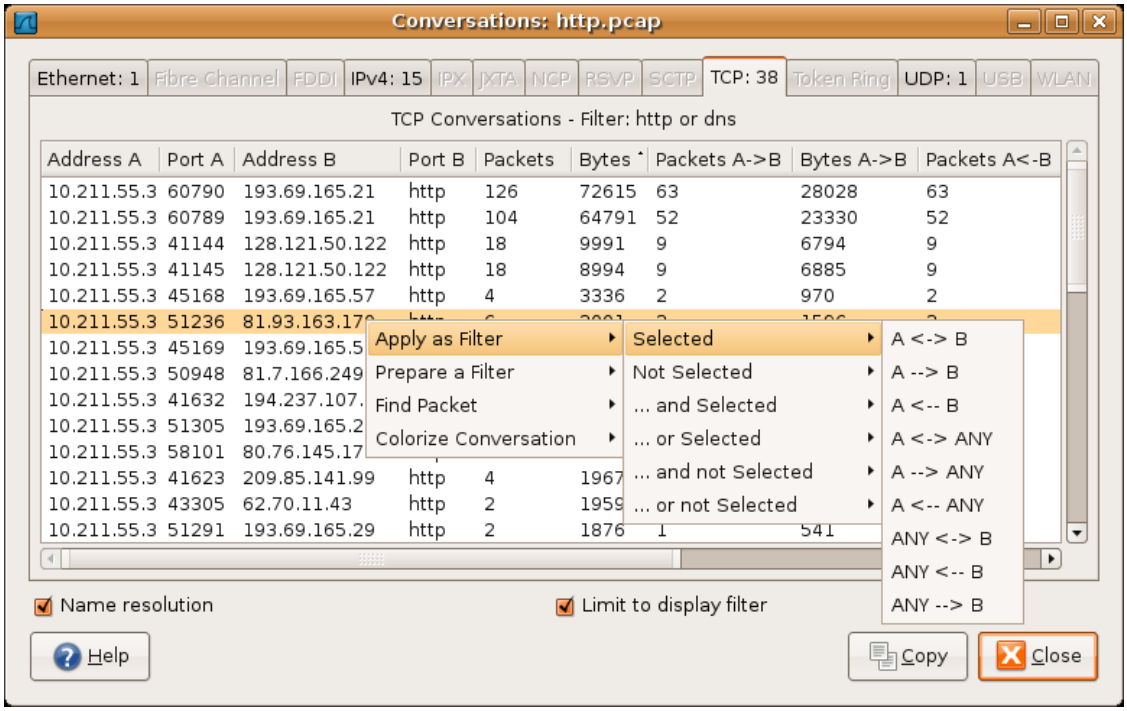
8.4.1. What is a Conversation?

A network conversation is the traffic between two specific endpoints. For example, an IP conversation is all the traffic between two IP addresses. The description of the known endpoint types can be found in Section 8.5.1, “What is an Endpoint?”.

8.4.2. The "Conversations" window

The conversations window is similar to the endpoint Window; see Section 8.5.2, “The "Endpoints" window” for a description of their common features. Along with addresses, packet counters, and byte counters the conversation window adds four columns: the time in seconds between the start of the capture and the start of the conversation ("Rel Start"), the duration of the conversation in seconds, and the average bits (not bytes) per second in each direction.

Figure 8.3. The "Conversations" window



Each row in the list shows the statistical values for exactly one conversation.

Name resolution will be done if selected in the window and if it is active for the specific protocol layer (MAC layer for the selected Ethernet endpoints page).

Limit to display filter will only show conversations matching the current display filter.

The **copy** button will copy the list values to the clipboard in CSV (Comma Separated Values) format.

Tip!

This window will be updated frequently, so it will be useful, even if you open it before (or while) you are doing a live capture.

8.4.3. The protocol specific "Conversation List" windows

Before the combined window described above was available, each of its pages was shown as a separate window. Even though the combined window is much more convenient to use, these separate windows are still available. The main reason is that they might process faster for very large capture files. However, as the functionality is exactly the same as in the combined window, they won't be discussed in detail here.

8.5. Endpoints

Statistics of the endpoints captured.

Tip!

If you are looking for a feature other network tools call a **hostlist**, here is the right place to look. The list of Ethernet or IP endpoints is usually what you're looking for.

8.5.1. What is an Endpoint?

A network endpoint is the logical endpoint of separate protocol traffic of a specific protocol layer. The endpoint statistics of Wireshark will take the following endpoints into account:

- **Ethernet**: an Ethernet endpoint is identical to the Ethernet's MAC address.
- **Fibre Channel**: XXX - insert info here.
- **FDDI**: a FDDI endpoint is identical to the FDDI MAC address.
- **IPv4**: an IP endpoint is identical to its IP address.
- **IPX**: XXX - insert info here.
- **TCP**: a TCP endpoint is a combination of the IP address and the TCP port used, so different TCP ports on the same IP address are different TCP endpoints.
- **Token Ring**: a Token Ring endpoint is identical to the Token Ring MAC address.
- **UDP**: a UDP endpoint is a combination of the IP address and the UDP port used, so different UDP ports on the same IP address are different UDP endpoints.

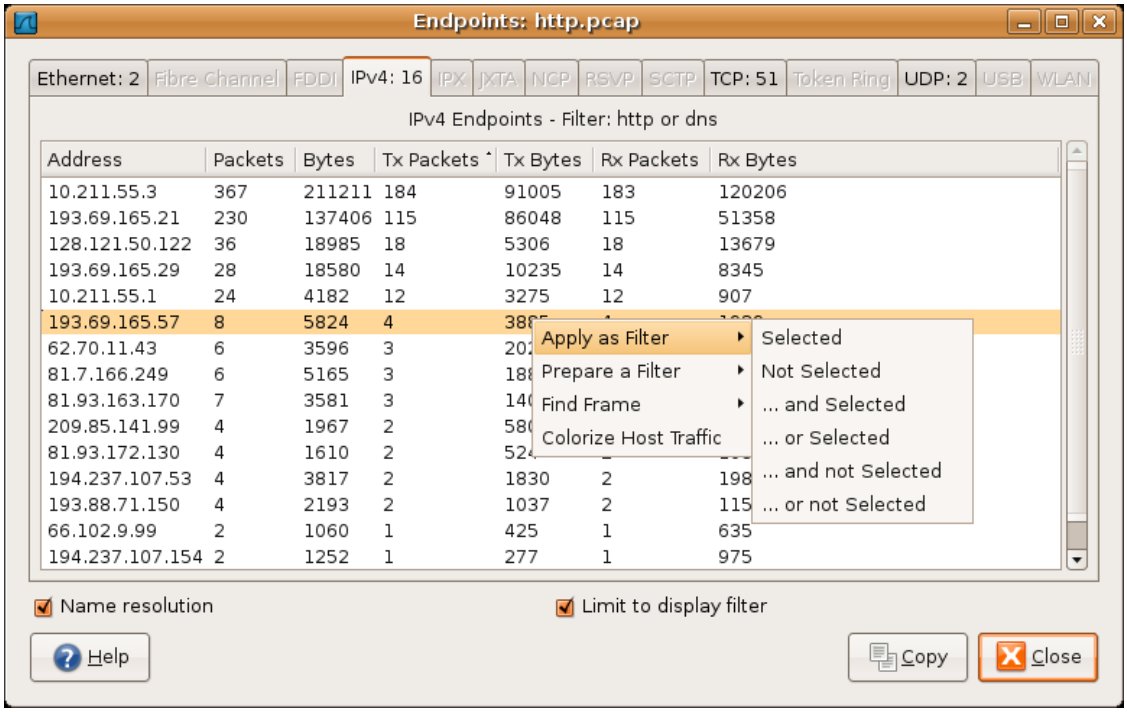
Broadcast / multicast endpoints

Broadcast / multicast traffic will be shown separately as additional endpoints. Of course, as these endpoints are virtual endpoints, the real traffic will be received by all (multicast: some) of the listed unicast endpoints.

8.5.2. The "Endpoints" window

This window shows statistics about the endpoints captured.

Figure 8.4. The "Endpoints" window



For each supported protocol, a tab is shown in this window. Each tab label shows the number of endpoints captured (e.g. the tab label "Ethernet: 5" tells you that five ethernet endpoints have been captured). If no endpoints of a specific protocol were captured, the tab label will be greyed out (although the related page can still be selected).

Each row in the list shows the statistical values for exactly one endpoint.

Name resolution will be done if selected in the window and if it is active for the specific protocol layer (MAC layer for the selected Ethernet endpoints page). As you might have noticed, the first row has a name resolution of the first three bytes "Netgear", the second row's address was resolved to an IP address (using ARP) and the third was resolved to a broadcast (unresolved this would still be: ff:ff:ff:ff:ff:ff); the last two Ethernet addresses remain unresolved.

Limit to display filter will only show conversations matching the current display filter.

The **copy** button will copy the list values to the clipboard in CSV (Comma Separated Values) format.

Tip!

This window will be updated frequently, so it will be useful, even if you open it before (or while) you are doing a live capture.

8.5.3. The protocol specific "Endpoint List" windows

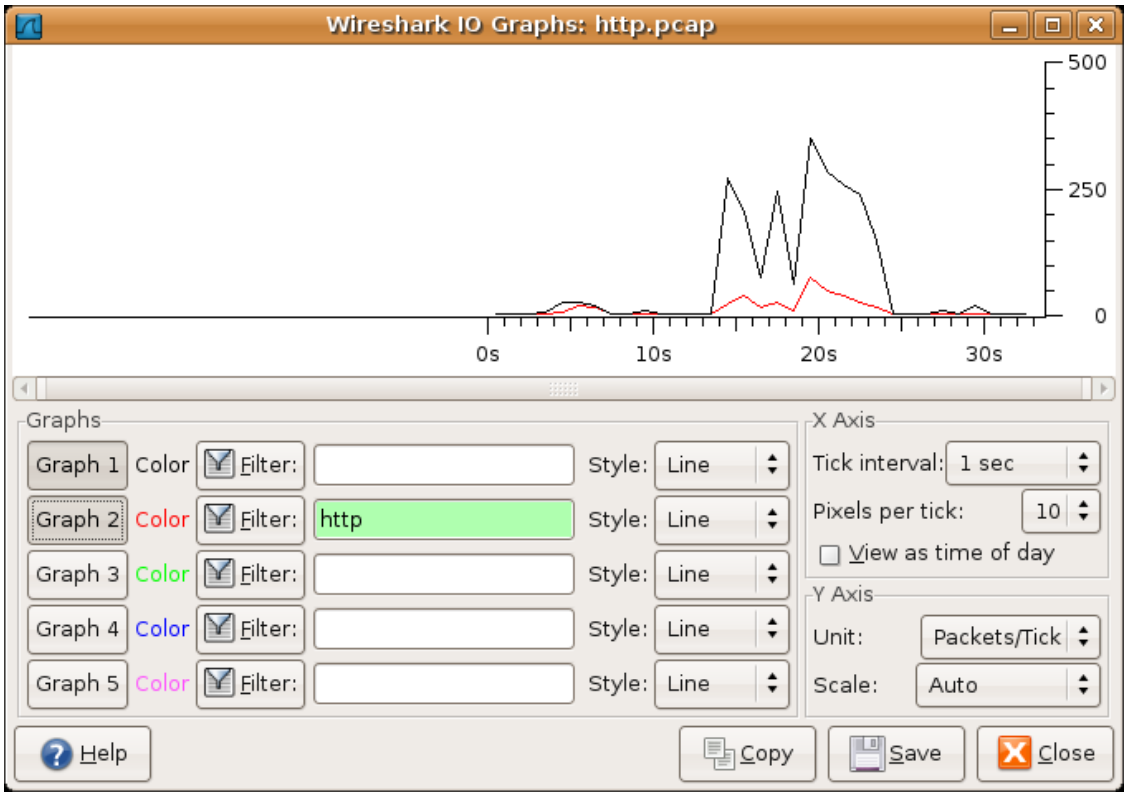
Before the combined window described above was available, each of its pages was shown as a separate window. Even though the combined window is much more convenient to use, these separate windows are still available. The main reason is that they might process faster for very large capture files. However, as the functionality is exactly the same as in the combined window, they won't be discussed in detail here.

8.6. The "IO Graphs" window

User configurable graph of the captured network packets.

You can define up to five differently colored graphs.

Figure 8.5. The "IO Graphs" window



The user can configure the following things:

- **Graphs**
  - **Graph 1-5**: enable the specific graph 1-5 (only graph 1 is enabled by default)
  - **Color**: the color of the graph (cannot be changed)
  - **Filter**: a display filter for this graph (only the packets that pass this filter will be taken into account for this graph)
  - **Style**: the style of the graph (Line/Impulse/FBar/Dot)
- **X Axis**
  - **Tick interval**: an interval in x direction lasts (10/1 minutes or 10/1/0.1/0.01/0.001 seconds)
  - **Pixels per tick**: use 10/5/2/1 pixels per tick interval

    - **View as time of day**: option to view x direction labels as time of day instead of seconds or minutes since beginning of capture

- **Y Axis**

    - **Unit**: the unit for the y direction (Packets/Tick, Bytes/Tick, Bits/Tick, Advanced...) [XXX - describe the Advanced feature.]

    - **Scale**: the scale for the y unit (Logarithmic,Auto,10,20,50,100,200,500,...)

The **save** button will save the currently displayed portion of the graph as one of various file formats. The save feature is only available when using GTK version 2.6 or higher (the latest Windows versions comply with this requirement) and Wireshark version 0.99.7 or higher.

The **copy** button will copy values from selected graphs to the clipboard in CSV (Comma Separated Values) format. The copy feature is only available in Wireshark version 0.99.8 or higher.

Tip!

Click in the graph to select the first package in the selected interval.

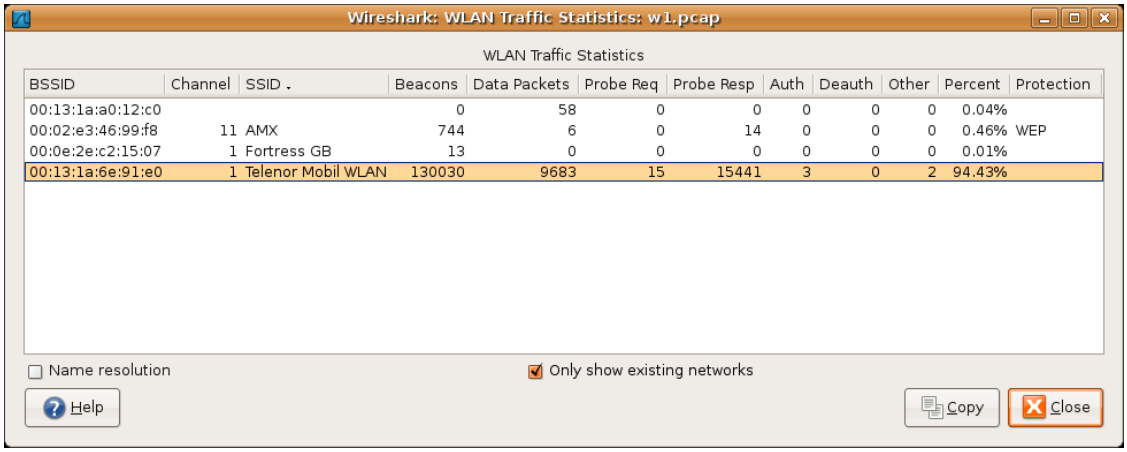
8.7. Compare two capture files

Compare two capture files.

8.8. WLAN Traffic Statistics

Statistics of the captured WLAN traffic. This window will summarize the wireless network traffic found in the capture. Probe requests will be merged into an existing network if the SSID matches.

Figure 8.6. The "WLAN Traffic Statistics" window


Wireshark: WLAN Traffic Statistics: w1.pcap
WLAN Traffic Statistics
BSSID Channel SSID Beacons Data Packets Probe Req Probe Resp Auth Deauth Other Percent Protection
00:13:1a:a0:12:c0 0 58 0 0 0 0 0 0.04%
00:02:e3:46:99:f8 11 AMX 744 6 0 14 0 0 0 0.46% WEP
00:0e:2e:c2:15:07 1 Fortress GB 13 0 0 0 0 0 0 0.01%
00:13:1a:6e:91:e0 1 Telenor Mobil WLAN 130030 9683 15 15441 3 0 2 94.43%
Name resolution
Only show existing networks
Help
Copy
Close

Each row in the list shows the statistical values for exactly one wireless network.

Name resolution will be done if selected in the window and if it is active for the MAC layer.

Only show existing networks will exclude probe requests with a SSID not matching any network from the list.

The **copy** button will copy the list values to the clipboard in CSV (Comma Separated Values) format.

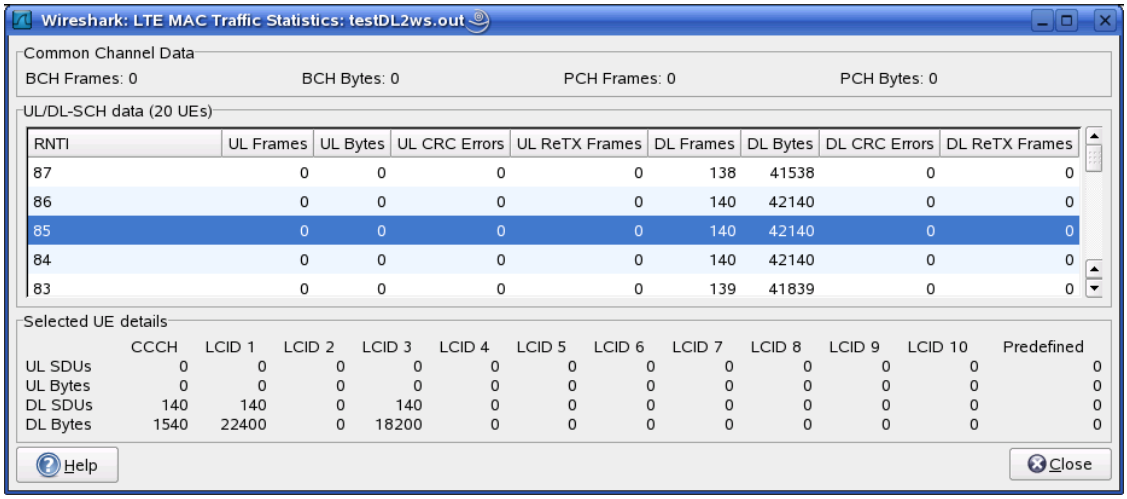
Tip!

This window will be updated frequently, so it will be useful, even if you open it before (or while) you are doing a live capture.

8.9. LTE MAC Traffic Statistics

Statistics of the captured LTE MAC traffic. This window will summarize the LTE MAC traffic found in the capture.

Figure 8.7. The "LTE MAC Traffic Statistics" window


Wireshark: LTE MAC Traffic Statistics: testDL2ws.out
Common Channel Data
BCH Frames: 0
BCH Bytes: 0
PCH Frames: 0
PCH Bytes: 0
UL/DL-SCH data (20 UEs)
RNTI UL Frames UL Bytes UL CRC Errors UL ReTX Frames DL Frames DL Bytes DL CRC Errors DL ReTX Frames
87 0 0 0 0 138 41538 0 0
86 0 0 0 0 140 42140 0 0
85 0 0 0 0 140 42140 0 0
84 0 0 0 0 140 42140 0 0
83 0 0 0 0 139 41839 0 0
Selected UE details
CCCH LCID 1 LCID 2 LCID 3 LCID 4 LCID 5 LCID 6 LCID 7 LCID 8 LCID 9 LCID 10 Predefined
UL SDUs 0 0 0 0 0 0 0 0 0 0 0 0
UL Bytes 0 0 0 0 0 0 0 0 0 0 0 0
DL SDUs 140 140 0 140 0 0 0 0 0 0 0 0
DL Bytes 1540 22400 0 18200 0 0 0 0 0 0 0 0
Help
Close

The top pane shows statistics for common channels. Each row in the middle pane shows statistical highlights for exactly one UE/C-RNTI. In the lower pane, you can see the for the currently selected UE/C-RNTI the traffic broken down by individual channel.

8.10. Service Response Time

The service response time is the time between a request and the corresponding response. This information is available for many protocols.

Service response time statistics are currently available for the following protocols:

- **DCE-RPC**
- **Fibre Channel**
- **H.225 RAS**
- **LDAP**
- **LTE MAC**
- **MGCP**
- **ONC-RPC**
- **SMB**

As an example, the DCE-RPC service response time is described in more detail.

Note!

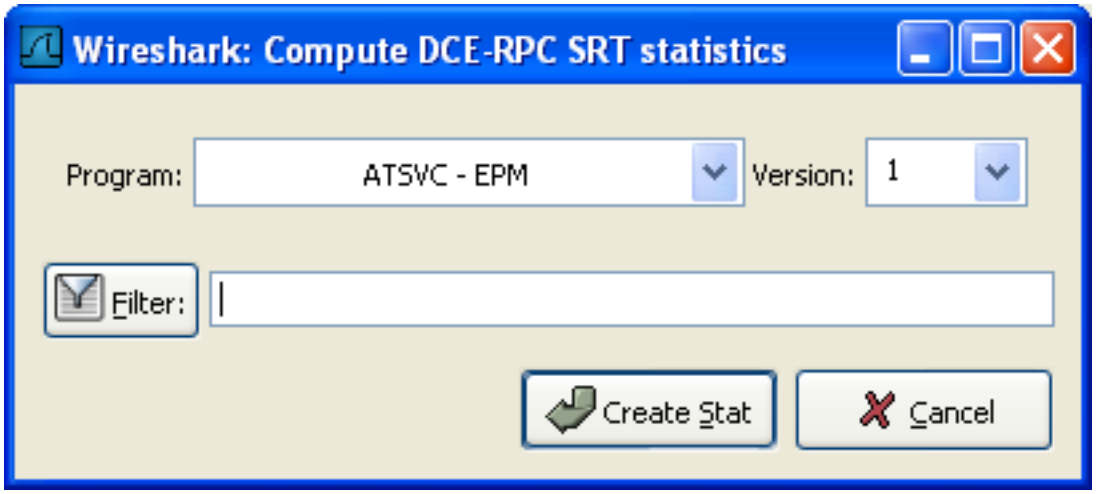
The other Service Response Time windows will work the same way (or only slightly different) compared to the following description.

8.10.1. The "Service Response Time DCE-RPC" window

The service response time of DCE-RPC is the time between the request and the corresponding response.

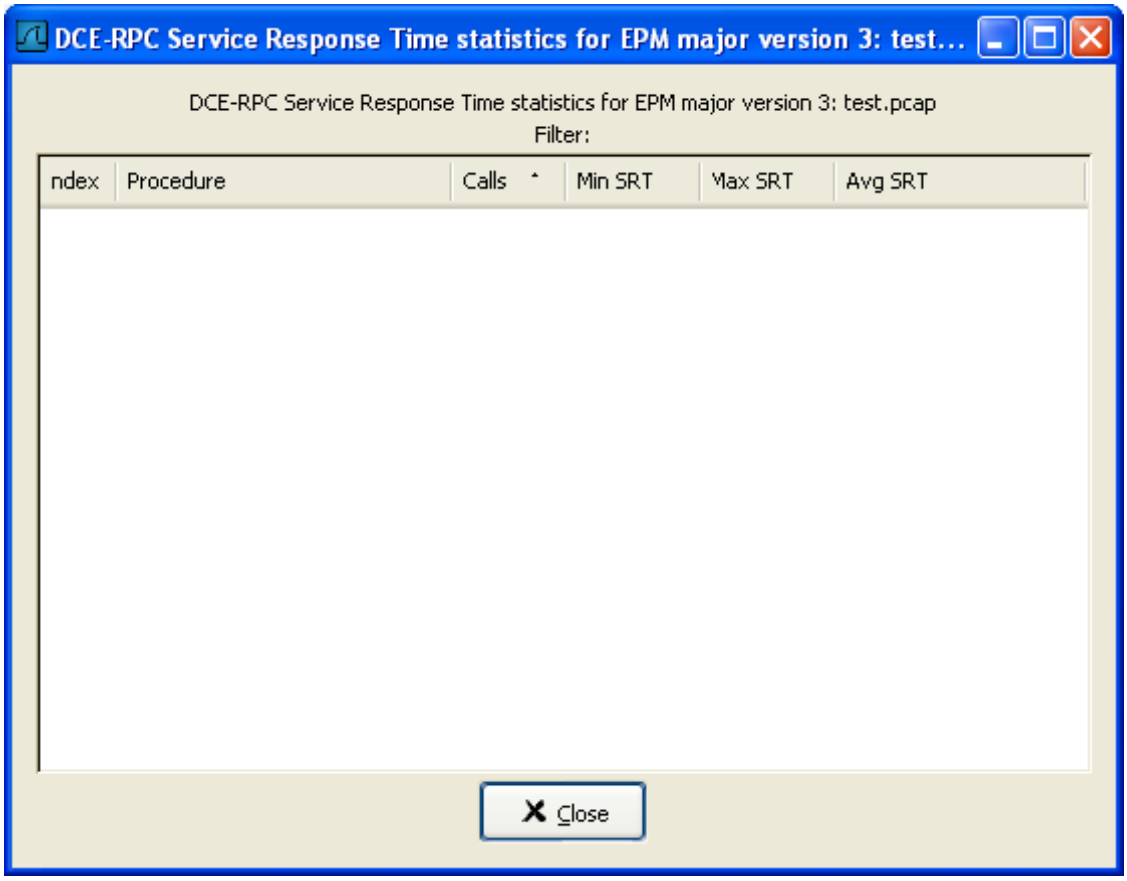
First of all, you have to select the DCE-RPC interface:

Figure 8.8. The "Compute DCE-RPC statistics" window



You can optionally set a display filter, to reduce the amount of packets.

Figure 8.9. The "DCE-RPC Statistic for ..." window



Each row corresponds to a method of the interface selected (so the EPM interface in version 3 has 7 methods). For each method the number of calls, and the statistics of the SRT time is calculated.

8.11. The protocol specific statistics windows

The protocol specific statistics windows display detailed information of specific protocols and might be described in a later version of this document.

Some of these statistics are described at the http://wiki.wireshark.org/Statistics pages.

Chapter 9. Customizing Wireshark

9.1. Introduction

Wireshark's default behaviour will usually suit your needs pretty well. However, as you become more familiar with Wireshark, it can be customized in various ways to suit your needs even better. In this chapter we explore:

- How to start Wireshark with command line parameters
- How to colorize the packet list
- How to control protocol dissection
- How to use the various preference settings

9.2. Start Wireshark from the command line

You can start Wireshark from the command line, but it can also be started from most Window managers as well. In this section we will look at starting it from the command line.

Wireshark supports a large number of command line parameters. To see what they are, simply enter the command **wireshark -h** and the help information shown in Example 9.1, "Help information available from Wireshark" (or something similar) should be printed.

Example 9.1. Help information available from Wireshark

```
Wireshark 0.99.6
Interactively dump and analyze network traffic.
See http://www.wireshark.org for more information.

Copyright 1998-2007 Gerald Combs <gerald@wireshark.org> and contributors.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Usage: wireshark [options] ... [ <infile> ]

Capture interface:
  -i <interface>           name or idx of interface (def: first non-loopback)
  -f <capture filter>      packet filter in libpcap filter syntax
  -s <snaplen>             packet snapshot length (def: 65535)
  -p                       don't capture in promiscuous mode
  -k                       start capturing immediately (def: do nothing)
  -Q                       quit Wireshark after capturing
  -S                       update packet display when new packets are captured
  -l                       turn on automatic scrolling while -S is in use
  -B <buffer size>         size of kernel buffer (def: 1MB)
  -y <link type>           link layer type (def: first appropriate)
  -D                       print list of interfaces and exit
  -L                       print list of link-layer types of iface and exit

Capture stop conditions:
  -c <packet count>        stop after n packets (def: infinite)
  -a <autostop cond.> ...  duration:NUM - stop after NUM seconds
                           filesize:NUM - stop this file after NUM KB
                              files:NUM - stop after NUM files
Capture output:
  -b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
                           filesize:NUM - switch to next file after NUM KB
                              files:NUM - ringbuffer: replace after NUM files
Input file:
  -r <infile>              set the filename to read from (no pipes or stdin!)

Processing:
  -R <read filter>         packet filter in Wireshark display filter syntax
  -n                       disable all name resolutions (def: all enabled)
  -N <name resolve flags>  enable specific name resolution(s): "mntC"

User interface:
  -g <packet number>       go to specified packet number after "-r"
  -m <font>                set the font name used for most text
  -t ad|a|r|d|dd|e         output format of time stamps (def: r: rel. to first)
  -X <key>:<value>         eXtension options, see man page for details
  -z <statistics>          show various statistics, see man page for details

Output:
  -w <outfile|->           set the output filename (or '-' for stdout)

Miscellaneous:
  -h                       display this help and exit
  -v                       display version info and exit
  -P <key:path>            persconf:path - personal configuration files
                           persdata:path - personal data files
  -o <name>:<value> ...    override preference or recent setting
```

We will examine each of the command line options in turn.

The first thing to notice is that issuing the command **wireshark** by itself will bring up Wireshark. However, you can include as many of the command line parameters as you like. Their meanings are as follows (in alphabetical order): XXX - is the alphabetical order a good choice? Maybe better task based?

-a <capture autostop condition> Specify a criterion that specifies when Wireshark is to stop writing to a capture file. The criterion is of the form test:value, where test is one of:

duration:value	Stop writing to a capture file after value of seconds have elapsed.
filesize:value	Stop writing to a capture file after it reaches a size of value kilobytes (where a kilobyte is 1000 bytes, not 1024 bytes). If this option is used together with the -b option, Wireshark will stop writing to the current capture file and switch to the next one if filesize is reached.
files:value	Stop writing to capture files after value number of files were written.

-b <capture ring buffer option> If a maximum capture file size was specified, this option causes Wireshark to run in "ring buffer" mode, with the specified number of files. In "ring buffer" mode, Wireshark will write to several capture files. Their name is based on the number of the file and on the creation date and time.

When the first capture file fills up, Wireshark will switch to writing to the next file, until it fills up the last file, at which point it'll discard the data in the first file (unless 0 is specified, in which case, the number of files is unlimited) and start writing to that file and so on.

If the optional duration is specified, Wireshark will also switch to the next file when the specified number of seconds has elapsed even if the current file is not completely fills up.

duration:value	Switch to the next file after value seconds have elapsed, even if the current file is not completely filled up.
filesize:value	Switch to the next file after it reaches a size of value kilobytes (where a kilobyte is 1000 bytes, not 1024 bytes).
files:value	Begin again with the first file after value number of files were written (form a ring buffer).

-B <capture buffer size (Win32 only)> Win32 only: set capture buffer size (in MB, default is 1MB). This is used by the the capture driver to buffer packet data until that data can be written to disk. If you encounter packet drops while capturing, try to increase this size.

-c <capture packet count>

This option specifies the maximum number of packets to capture when capturing live data. It would be used in conjunction with the **-k** option.

-D

Print a list of the interfaces on which Wireshark can capture, and exit. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the **-i** flag to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (e.g., Windows systems, or UNIX systems lacking **ifconfig -a**); the number can be useful on Windows 2000 and later systems, where the interface name is a somewhat complex string.

Note that "can capture" means that Wireshark was able to open that device to do a live capture; if, on your system, a program doing a network capture must be run from an account with special privileges (for example, as root), then, if Wireshark is run with the **-D** flag and is not run from such an account, it will not list any interfaces.

-f <capture filter>

This option sets the initial capture filter expression to be used when capturing packets.

-g <packet number>

After reading in a capture file using the -r flag, go to the given packet number.

-h

The **-h** option requests Wireshark to print its version and usage instructions (as shown above) and exit.

-i <capture interface>

Set the name of the network interface or pipe to use for live packet capture.

Network interface names should match one of the names listed in **wireshark -D** (described above); a number, as reported by **wireshark -D**, can also be used. If you're using UNIX, **netstat -i** or **ifconfig -a** might also work to list interface names, although not all versions of UNIX support the **-a** flag to **ifconfig**.

If no interface is specified, Wireshark searches the list of interfaces, choosing the first non-loopback interface if there are any non-loopback interfaces, and choosing the first loopback interface if there are no non-loopback interfaces; if there are no interfaces, Wireshark reports an error and doesn't start the capture.

Pipe names should be either the name of a FIFO (named pipe) or ``-'' to read data from the standard input. Data read from pipes must be in standard libpcap format.

-k

The **-k** option specifies that Wireshark should start capturing packets immediately. This option requires the use of the **-i** parameter to specify the interface that packet capture will occur from.

-l

This option turns on automatic scrolling if the packet list pane is being updated automatically as packets arrive during a capture (as specified by the **-S** flag).

-L

List the data link types supported by the interface and exit.

-m <font>

This option sets the name of the font used for most text displayed by Wireshark. XXX - add an example!

-n

Disable network object name resolution (such as hostname, TCP and UDP port names).

-N <name resolving flags>

Turns on name resolving for particular types of addresses and port numbers; the argument is a string that may contain the letters **m** to enable MAC address resolution, **n** to enable network address resolution, and **t** to enable transport-layer port number resolution. This overrides **-n** if both **-N** and **-n** are present. The letter C enables concurrent (asynchronous) DNS lookups.

-o <preference/recent settings>

Sets a preference or recent value, overriding the default value and any value read from a preference/recent file. The argument to the flag is a string of the form prefname:value, where prefname is the name of the preference (which is the same name that would appear in the preference/recent file), and value is the value to which it should be set. Multiple instances of **-o <preference settings>** can be given on a single command line.

An example of setting a single preference would be:

wireshark -o mgcp.display_dissect_tree:TRUE

An example of setting multiple preferences would be:

wireshark -o mgcp.display_dissect_tree:TRUE -o mgcp.udp.callagent_port:2627

Tip!

You can get a list of all available preference strings from the preferences file, see Appendix A, *Files and Folders*.

User access tables can be overridden using "uat," followed by the UAT file name and a valid record for the file:

wireshark -o "uat:user_dlts:\"User 0 (DLT=147)\",\"http\",\"0\",\"\",\"0\",\"\""

The example above would dissect packets with a libpcap data link type 147 as HTTP, just as if you had configured it in the DLT_USER protocol preferences.

-p

Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, -p cannot be used to ensure that the only traffic that is captured is traffic sent to or from the machine on which Wireshark is running, broadcast traffic, and multicast traffic to addresses received by that machine.

-P <path setting>

Special path settings usually detected automatically. This is used for special cases, e.g. starting Wireshark from a known location on an USB stick.

The criterion is of the form key:path, where key is one of:

persconf:path — path of personal configuration files, like the preferences files.

persdata:path — path of personal data files, it's the folder initially opened. After the initialization, the recent file will keep the folder last used.

-Q — This option forces Wireshark to exit when capturing is complete. It can be used with the **-c** option. It must be used in conjunction with the **-i** and **-w** options.

-r <infile> — This option provides the name of a capture file for Wireshark to read and display. This capture file can be in one of the formats Wireshark understands.

-R <read (display) filter> — This option specifies a display filter to be applied when reading packets from a capture file. The syntax of this filter is that of the display filters discussed in Section 6.3, “Filtering packets while viewing”. Packets not matching the filter are discarded.

-s <capture snaplen> — This option specifies the snapshot length to use when capturing packets. Wireshark will only capture **<snaplen>** bytes of data for each packet.

-S — This option specifies that Wireshark will display packets as it captures them. This is done by capturing in one process and displaying them in a separate process. This is the same as "Update list of packets in real time" in the Capture Options dialog box.

-t <time stamp format> — This option sets the format of packet timestamps that are displayed in the packet list window. The format can be one of:

- **r** relative, which specifies timestamps are displayed relative to the first packet captured.
- **a** absolute, which specifies that actual times be displayed for all packets.
- **ad** absolute with date, which specifies that actual dates and times be displayed for all packets.
- **d** delta, which specifies that timestamps are relative to the previous packet.
- **e** epoch, which specifies that timestamps are seconds since epoch (Jan 1, 1970 00:00:00)

-v — The **-v** option requests Wireshark to print out its version information and exit.

-w <savefile> — This option sets the name of the **savefile** to be used when saving a capture file.

-y <capture link type> — If a capture is started from the command line with -k, set the data link type to use while capturing packets. The values reported by -L are the values that can be used.

-X <eXtension option>

Specify an option to be passed to a TShark module. The eXtension option is in the form extension_key:value, where extension_key can be:

lua_script:lua_script_filename; Tells Wireshark to load the given script in addition to the default Lua scripts.

-z <statistics-string>

Get Wireshark to collect various types of statistics and display the result in a window that updates in semi-real time. XXX - add more details here!

9.3. Packet colorization

A very useful mechanism available in Wireshark is packet colorization. You can set-up Wireshark so that it will colorize packets according to a filter. This allows you to emphasize the packets you are (usually) interested in.

Tip!

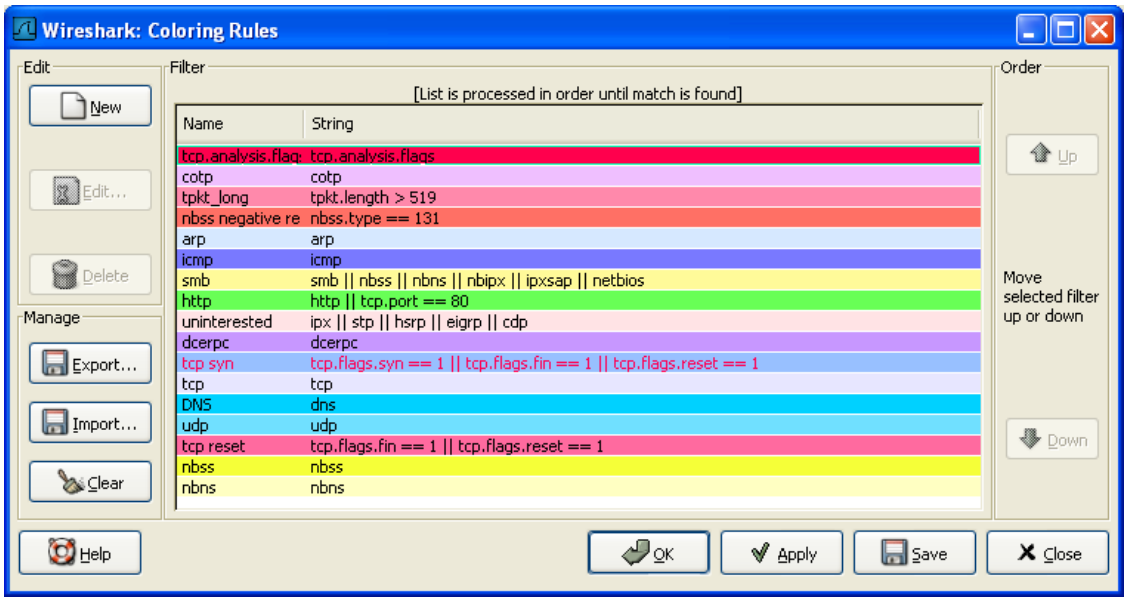
You will find a lot of Coloring Rule examples at the **Wireshark Wiki Coloring Rules page** at http://wiki.wireshark.org/ColoringRules.

There are two types of coloring rules in Wireshark. Temporary ones that are only used until you quit the program. And permanent ones that will be saved to a preference file so that they are available on a next session.

Temporary coloring rules can be added by selecting a packet and pressing the <ctrl> key together with one of the number keys. This will create a coloring rule based on the currently selected conversation. It will try to create a conversation filter based on TCP first, then UDP, then IP and at last Ethernet. Temporary filters can also be created by selecting the "Colorize with Filter > Color X" menu items when rightclicking in the packet-detail pane.

To permanently colorize packets, select the **Coloring Rules...** menu item from the **View** menu; Wireshark will pop up the "Coloring Rules" dialog box as shown in Figure 9.1, “The "Coloring Rules" dialog box”.

Figure 9.1. The "Coloring Rules" dialog box



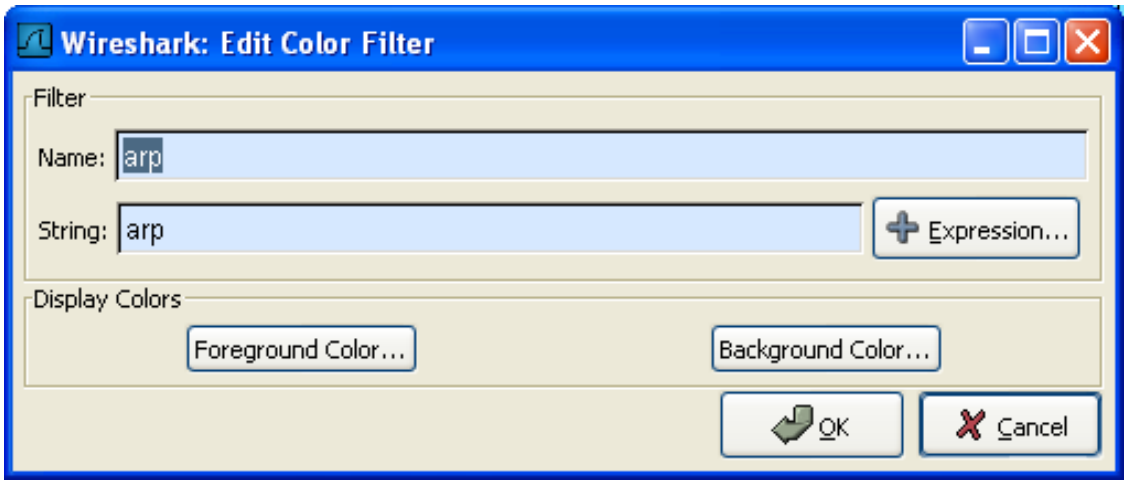
Once the Coloring Rules dialog box is up, there are a number of buttons you can use, depending on whether or not you have any color filters installed already.

Note!

You will need to carefully select the order the coloring rules are listed as they are applied in order from top to bottom. So, more specific rules need to be listed before more general rules. For example, if you have a color rule for UDP before the one for DNS, the color rule for DNS will never be applied (as DNS uses UDP, so the UDP rule will match first).

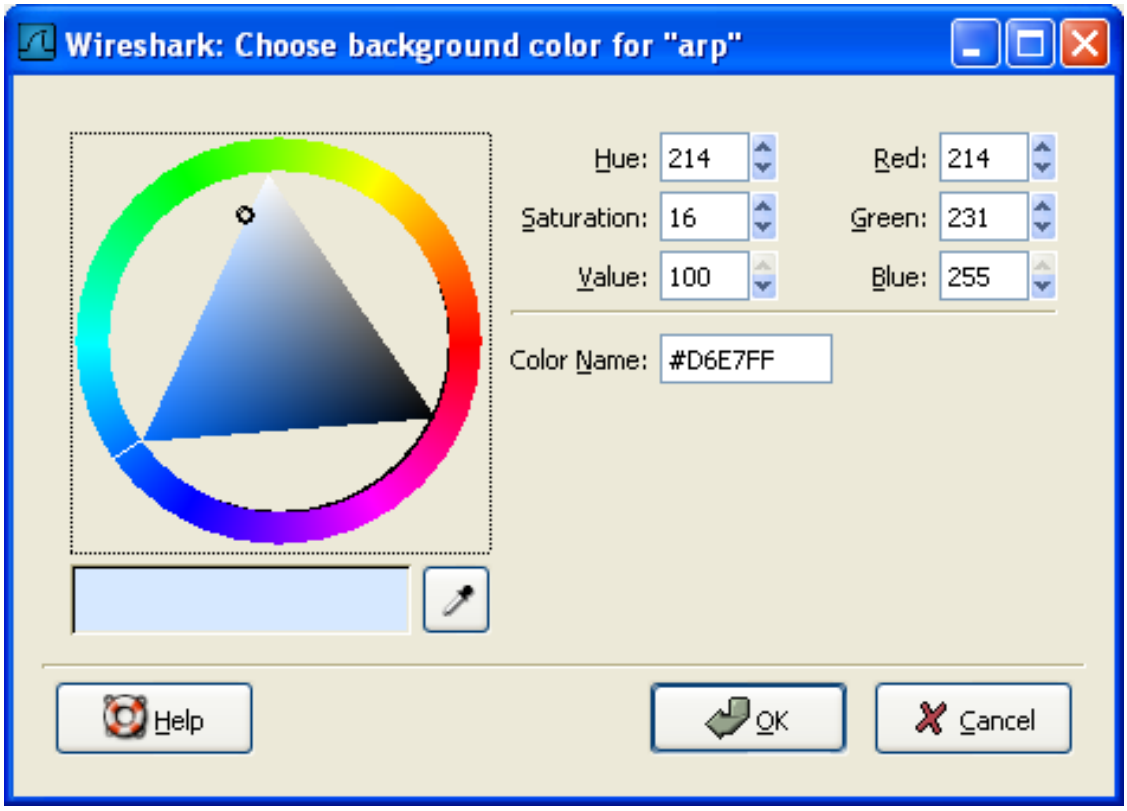
If this is the first time you have used Coloring Rules, click on the New button which will bring up the Edit color filter dialog box as shown in Figure 9.2, “The "Edit Color Filter" dialog box”.

Figure 9.2. The "Edit Color Filter" dialog box



In the Edit Color dialog box, simply enter a name for the color filter, and enter a filter string in the Filter text field. Figure 9.2, “The "Edit Color Filter" dialog box” shows the values **arp** and **arp** which means that the name of the color filter is **arp** and the filter will select protocols of type **arp**. Once you have entered these values, you can choose a foreground and background color for packets that match the filter expression. Click on **Foreground color...** or **Background color...** to achieve this and Wireshark will pop up the Choose foreground/background color for protocol dialog box as shown in Figure 9.3, “The "Choose color" dialog box”.

Figure 9.3. The "Choose color" dialog box



Select the color you desire for the selected packets and click on OK.

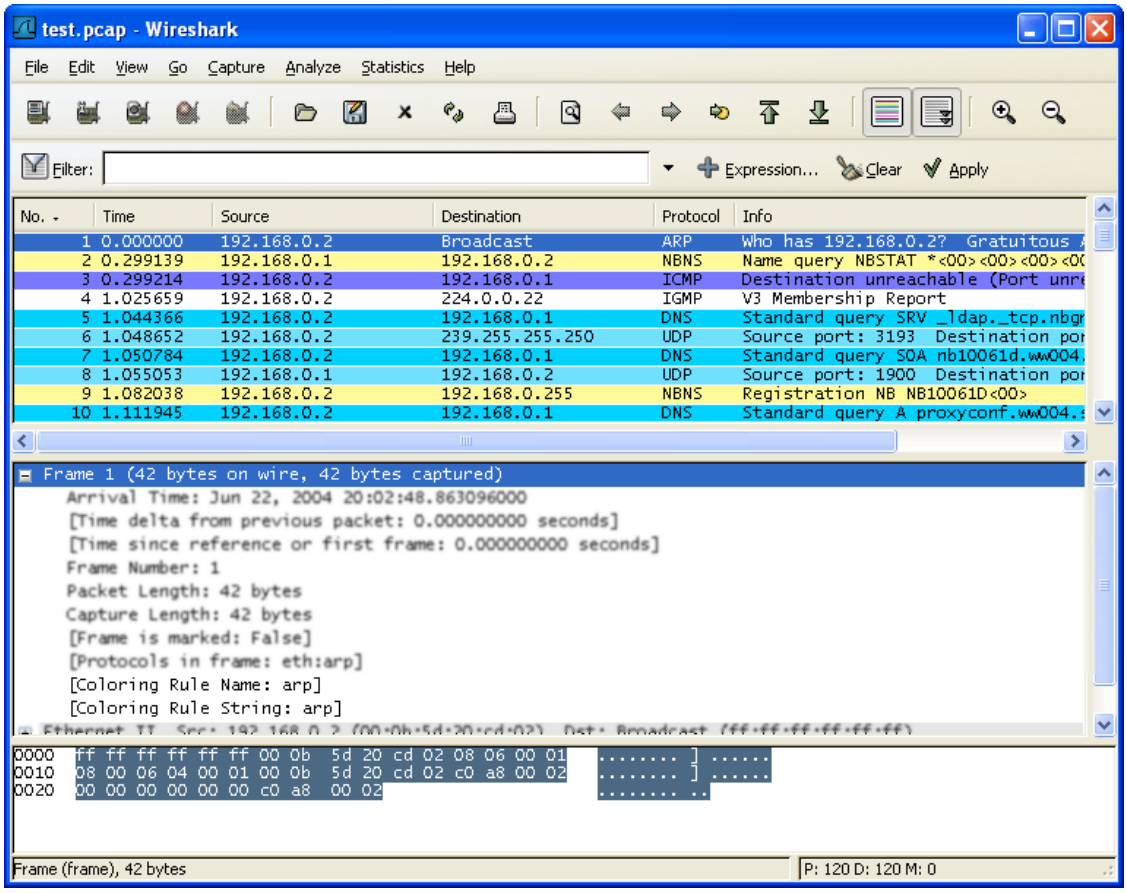
Note!

You must select a color in the colorbar next to the colorwheel to load values into the RGB values. Alternatively, you can set the values to select the color you want.

Figure 9.4, “Using color filters with Wireshark” shows an example of several color filters being used in Wireshark. You may not like the color choices, however, feel free to choose your own.

If you are uncertain which coloring rule actually took place for a specific packet, have a look at the [Coloring Rule Name: ...] and [Coloring Rule String: ...] fields.

Figure 9.4. Using color filters with Wireshark


test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
1 0.000000 192.168.0.2 Broadcast ARP Who has 192.168.0.2? Gratuitous
2 0.299139 192.168.0.1 192.168.0.2 NBNS Name query NBSTAT *<00><00><00>
3 0.299214 192.168.0.2 192.168.0.1 ICMP Destination unreachable (Port unr
4 1.025659 192.168.0.2 224.0.0.22 IGMP V3 Membership Report
5 1.044366 192.168.0.2 192.168.0.1 DNS Standard query SRV _ldap._tcp.nbg
6 1.048652 192.168.0.2 239.255.255.250 UDP Source port: 3193 Destination po
7 1.050784 192.168.0.2 192.168.0.1 DNS Standard query SOA nb10061d.ww004.
8 1.055053 192.168.0.1 192.168.0.2 UDP Source port: 1900 Destination po
9 1.082038 192.168.0.2 192.168.0.255 NBNS Registration NB NB10061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS Standard query A proxyconf.ww004.
Frame 1 (42 bytes on wire, 42 bytes captured)
Arrival Time: Jun 22, 2004 20:02:48.863096000
[Time delta from previous packet: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Packet Length: 42 bytes
Capture Length: 42 bytes
[Frame is marked: False]
[Protocols in frame: eth:arp]
[Coloring Rule Name: arp]
[Coloring Rule String: arp]
0000 ff ff ff ff ff ff 00 0b 5d 20 cd 02 08 06 00 01
0010 08 00 06 04 00 01 00 0b 5d 20 cd 02 c0 a8 00 02
0020 00 00 00 00 00 00 c0 a8 00 02
Frame (frame), 42 bytes
P: 120 D: 120 M: 0

9.4. Control Protocol dissection

The user can control how protocols are dissected.

Each protocol has its own dissector, so dissecting a complete packet will typically involve several dissectors. As Wireshark tries to find the right dissector for each packet (using static "routes" and heuristics "guessing"), it might choose the wrong dissector in your specific case. For example, Wireshark won't know if you use a common protocol on an uncommon TCP port, e.g. using HTTP on TCP port 800 instead of the standard port 80.

There are two ways to control the relations between protocol dissectors: disable a protocol dissector completely or temporarily divert the way Wireshark calls the dissectors.

9.4.1. The "Enabled Protocols" dialog box

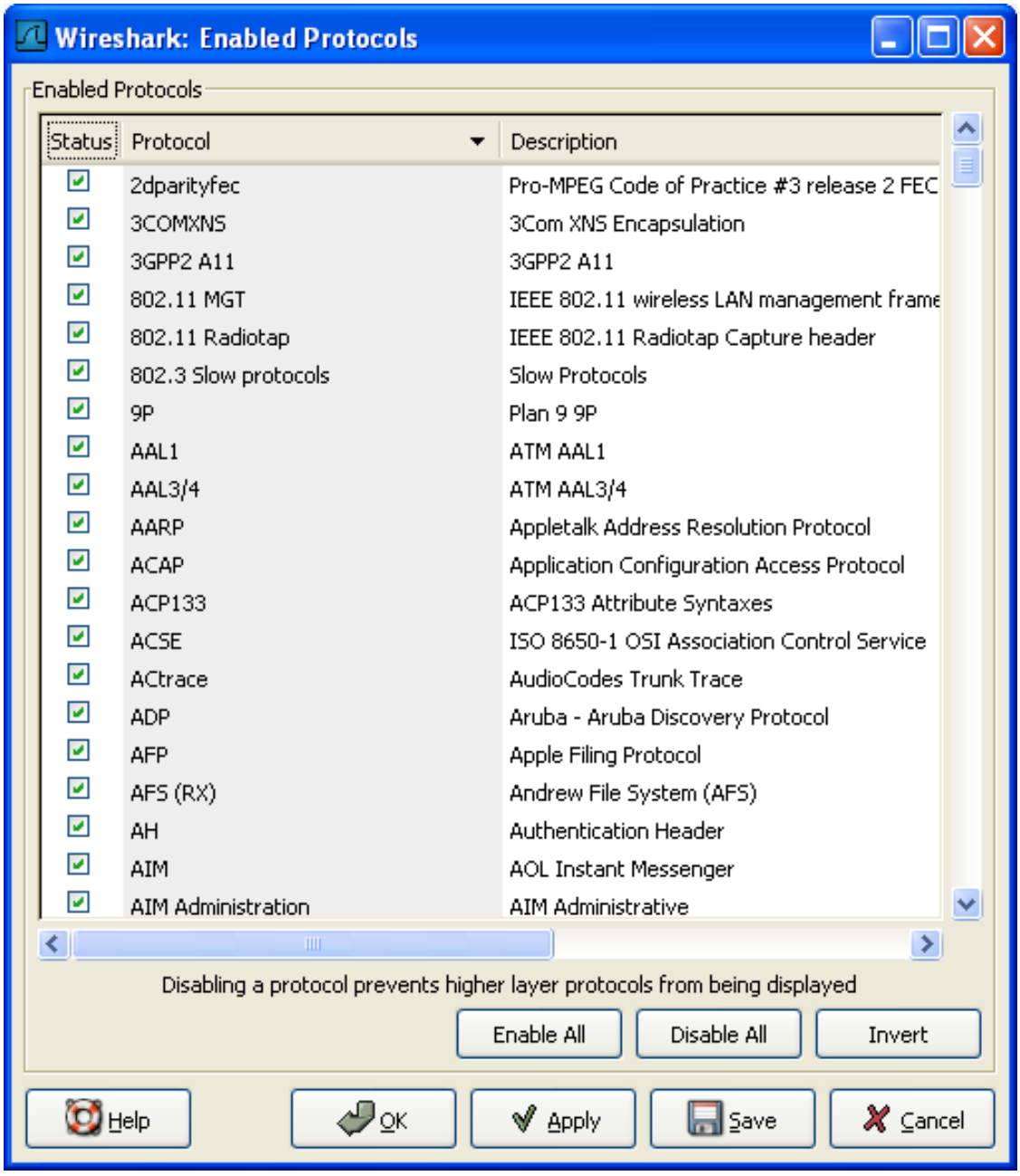
The Enabled Protocols dialog box lets you enable or disable specific protocols; all protocols are enabled by default. When a protocol is disabled, Wireshark stops processing a packet whenever that protocol is encountered.

Note!

Disabling a protocol will prevent information about higher-layer protocols from being displayed. For example, suppose you disabled the IP protocol and selected a packet containing Ethernet, IP, TCP, and HTTP information. The Ethernet information would be displayed, but the IP, TCP and HTTP information would not - disabling IP would prevent it and the other protocols from being displayed.

To enable/disable protocols select the **Enabled Protocols...** item from the **Analyze** menu; Wireshark will pop up the "Enabled Protocols" dialog box as shown in Figure 9.5, “The "Enabled Protocols" dialog box”.

Figure 9.5. The "Enabled Protocols" dialog box



To disable or enable a protocol, simply click on it using the mouse or press the space bar when the protocol is highlighted. Note that typing the first few letters of the protocol name when the Enabled Protocols dialog box is active will temporarily open a search text box and automatically select the first matching protocol name (if it exists).

Warning!

You have to use the Save button to save your settings. The OK or Apply buttons will not save your changes permanently, so they will be lost when Wireshark is closed.

You can choose from the following actions:

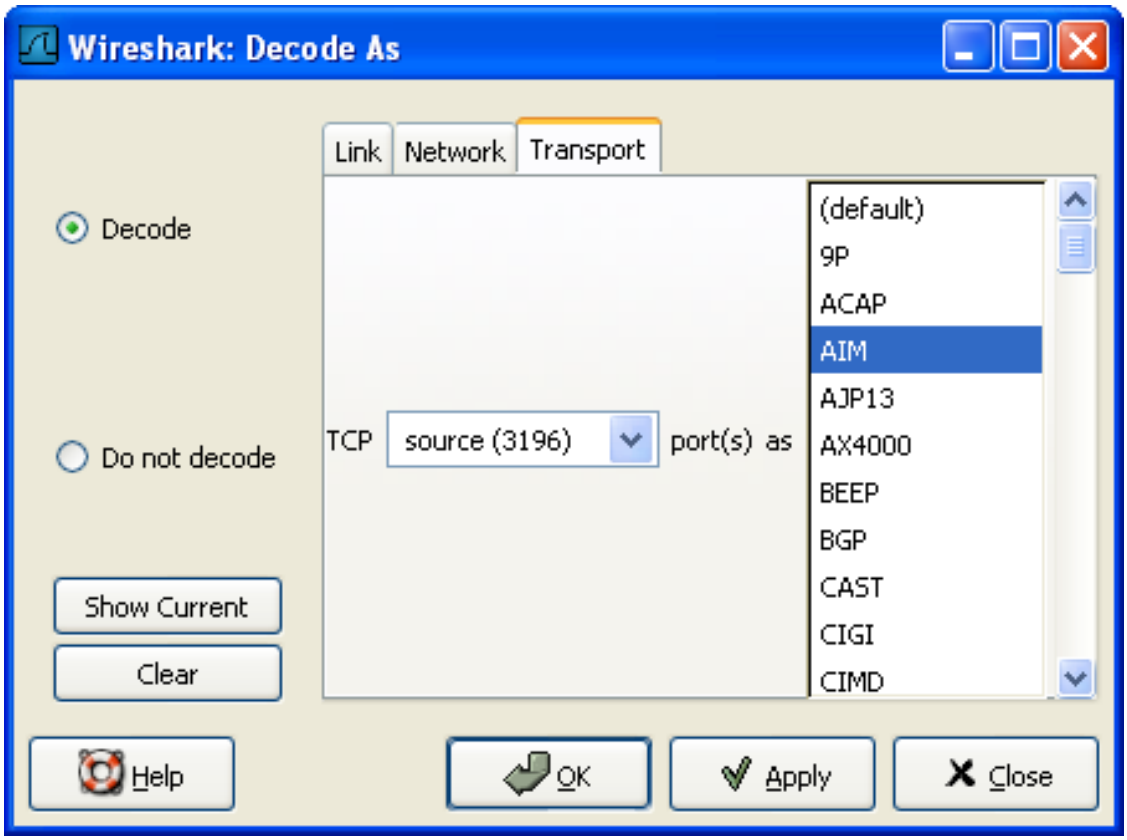
1. **Enable All**: Enable all protocols in the list.
2. **Disable All**: Disable all protocols in the list.
3. **Invert**: Toggle the state of all protocols in the list.
4. **OK**: Apply the changes and close the dialog box.
5. **Apply**: Apply the changes and keep the dialog box open.
6. **Save**: Save the settings to the disabled_protos, see Appendix A, *Files and Folders* for details.
7. **Cancel**: Cancel the changes and close the dialog box.

9.4.2. User Specified Decodes

The "Decode As" functionality let you temporarily divert specific protocol dissections. This might be useful for example, if you do some uncommon experiments on your network.

Decode As is accessed by selecting the **Decode As...** item from the **Analyze** menu; Wireshark will pop up the "Decode As" dialog box as shown in Figure 9.6, “The "Decode As" dialog box”.

Figure 9.6. The "Decode As" dialog box



The content of this dialog box depends on the selected packet when it was opened.

Warning!

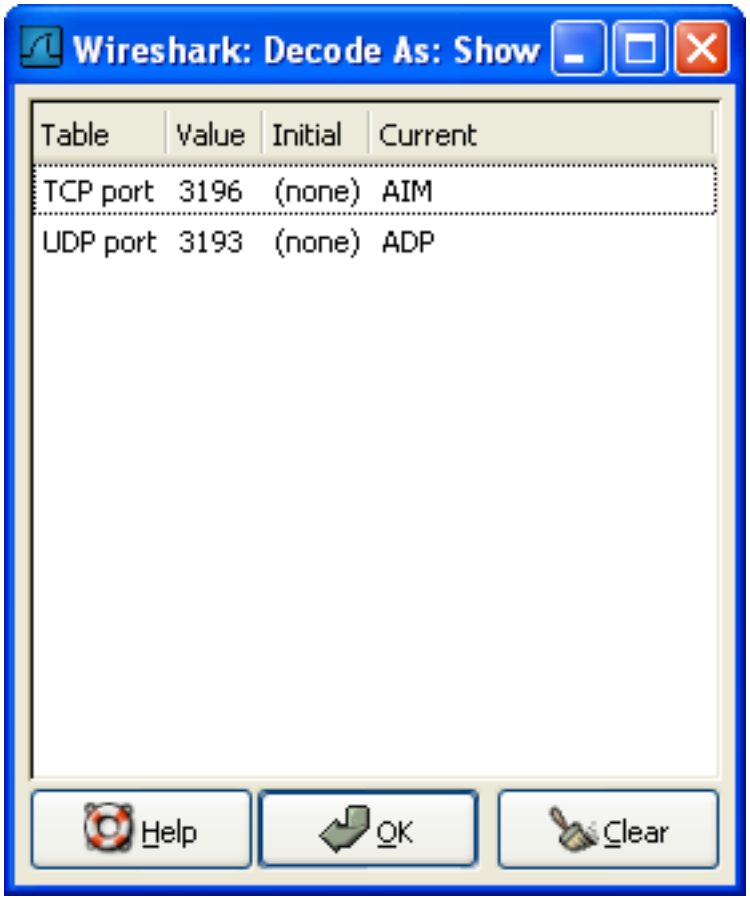
The user specified decodes can not be saved. If you quit Wireshark, these settings will be lost.

1. **Decode**: Decode packets the selected way.
2. **Do not decode**: Do not decode packets the selected way.
3. **Link/Network/Transport**: Specify the network layer at which "Decode As" should take place. Which of these pages are available depends on the content of the selected packet when this dialog box is opened.
4. **Show Current**: Open a dialog box showing the current list of user specified decodes.
5. **OK**: Apply the currently selected decode and close the dialog box.
6. **Apply**: Apply the currently selected decode and keep the dialog box open.
7. **Cancel**: Cancel the changes and close the dialog box.

9.4.3. Show User Specified Decodes

This dialog box shows the currently active user specified decodes.

Figure 9.7. The "Decode As: Show" dialog box



1. **OK**: Close this dialog box.
2. **Clear**: Removes all user specified decodes.

9.5. Preferences

There are a number of preferences you can set. Simply select the **Preferences...** menu item from the **Edit** menu; and Wireshark will pop up the Preferences dialog box as shown in Figure 9.8, “The preferences dialog box”, with the "User Interface" page as default. On the left side is a tree where you can select the page to be shown.

Note!

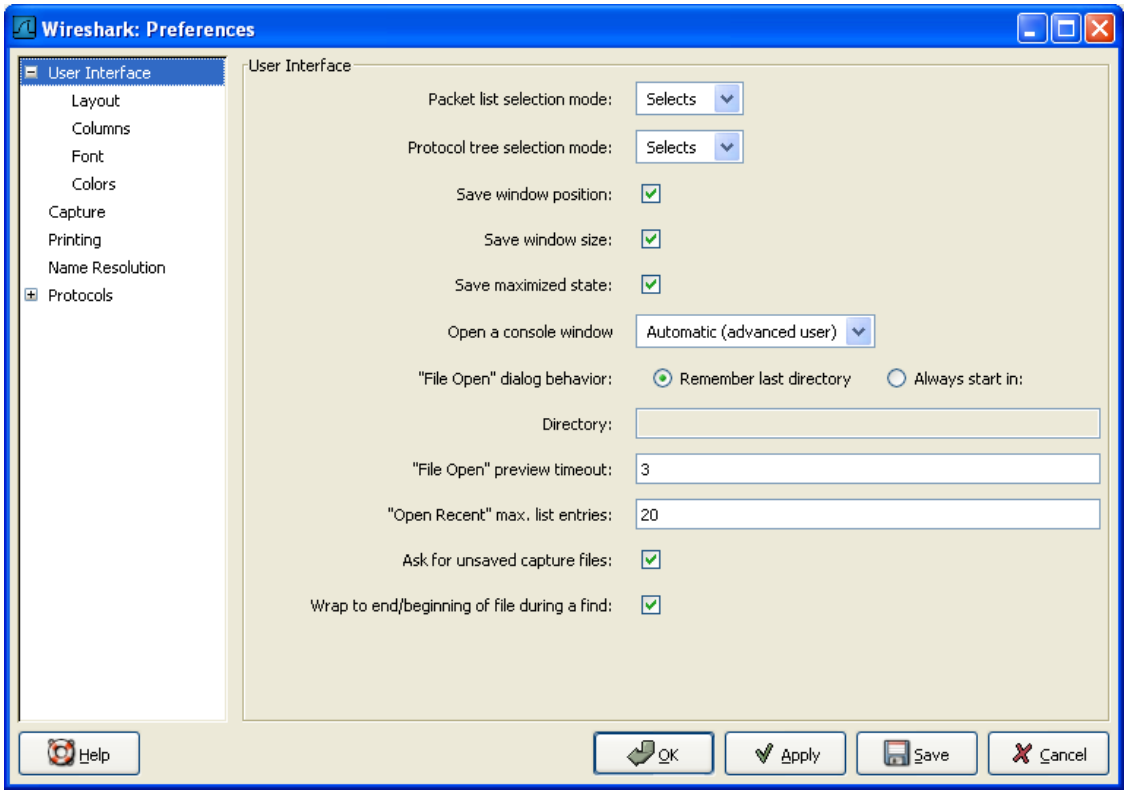
Preference settings are added frequently. For a recent explanation of the preference pages and their settings have a look at the **Wireshark Wiki Preferences page** at http://wiki.wireshark.org/Preferences.

Warning!

The OK or Apply button will not save the preference settings, you'll have to save the settings by clicking the Save button.

- The **OK** button will apply the preferences settings and close the dialog.
- The **Apply** button will apply the preferences settings and keep the dialog open.
- The **Save** button will apply the preferences settings, save the settings on the hard disk and keep the dialog open.
- The **Cancel** button will restore all preferences settings to the last saved state.

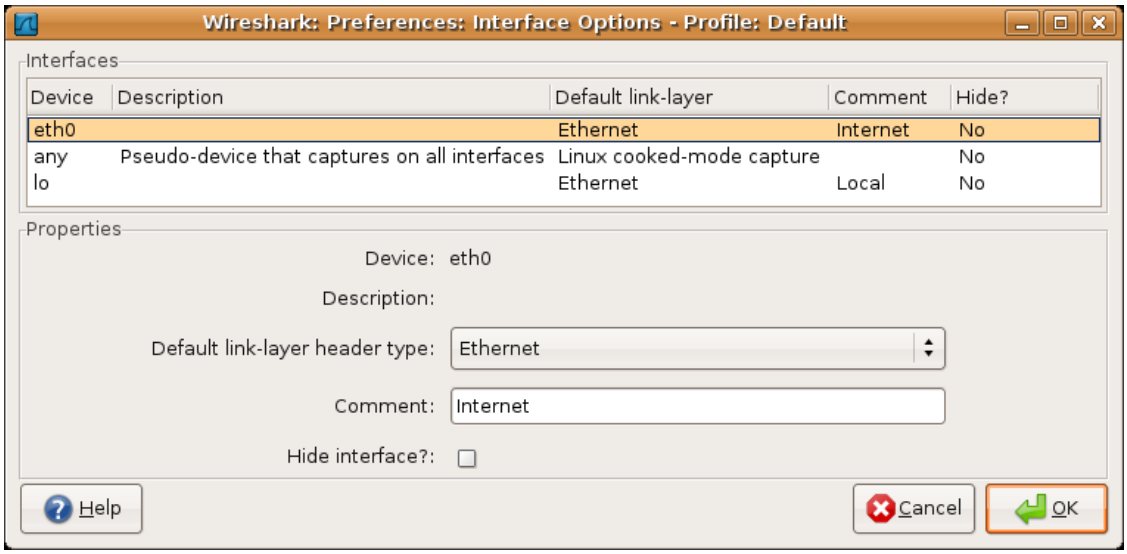
Figure 9.8. The preferences dialog box



9.5.1. Interface Options

In the Capture preferences it is possible to configure several options for the interfaces available on your computer. Select the **Capture** pane and press the Interfaces: **Edit** button. In this window it is possible to change the default link-layer header type for the interface, add a comment or choose to hide a interface from other parts of the program.

Figure 9.9. The interface options dialog box



Each row contains options for each interface available on your computer.

- **Device**: the device name provided by the operating system.
- **Description**: provided by the operating system.
- **Default link-layer**: each interface may provide several link-layer header types. The default link-layer chosen here is the one used when you first start Wireshark. It is also possible to change this value in Section 4.5, “The "Capture Options" dialog box” when you start a capture. For a detailed description, see Section 4.8, “Link-layer header type”.
- **Comment**: a user provided description of the interface. This comment will be used as a description instead of the operating system description.
- **Hide?**: enable this option to hide the interface from other parts of the program.

9.6. Configuration Profiles

Configuration Profiles can be used to configure and use more than one set of preferences and configurations. Select the **Configuration Profiles...** menu item from the **Edit** menu, or simply press Shift-Ctrl-A; and Wireshark will pop up the Configuration Profiles dialog box as shown in Figure 9.10, “The configuration profiles dialog box”. It is also possible to click in the "Profile" part of the statusbar to popup a menu with available Configuration Profiles (Figure 3.19, “The Statusbar with a configuration profile menu”).

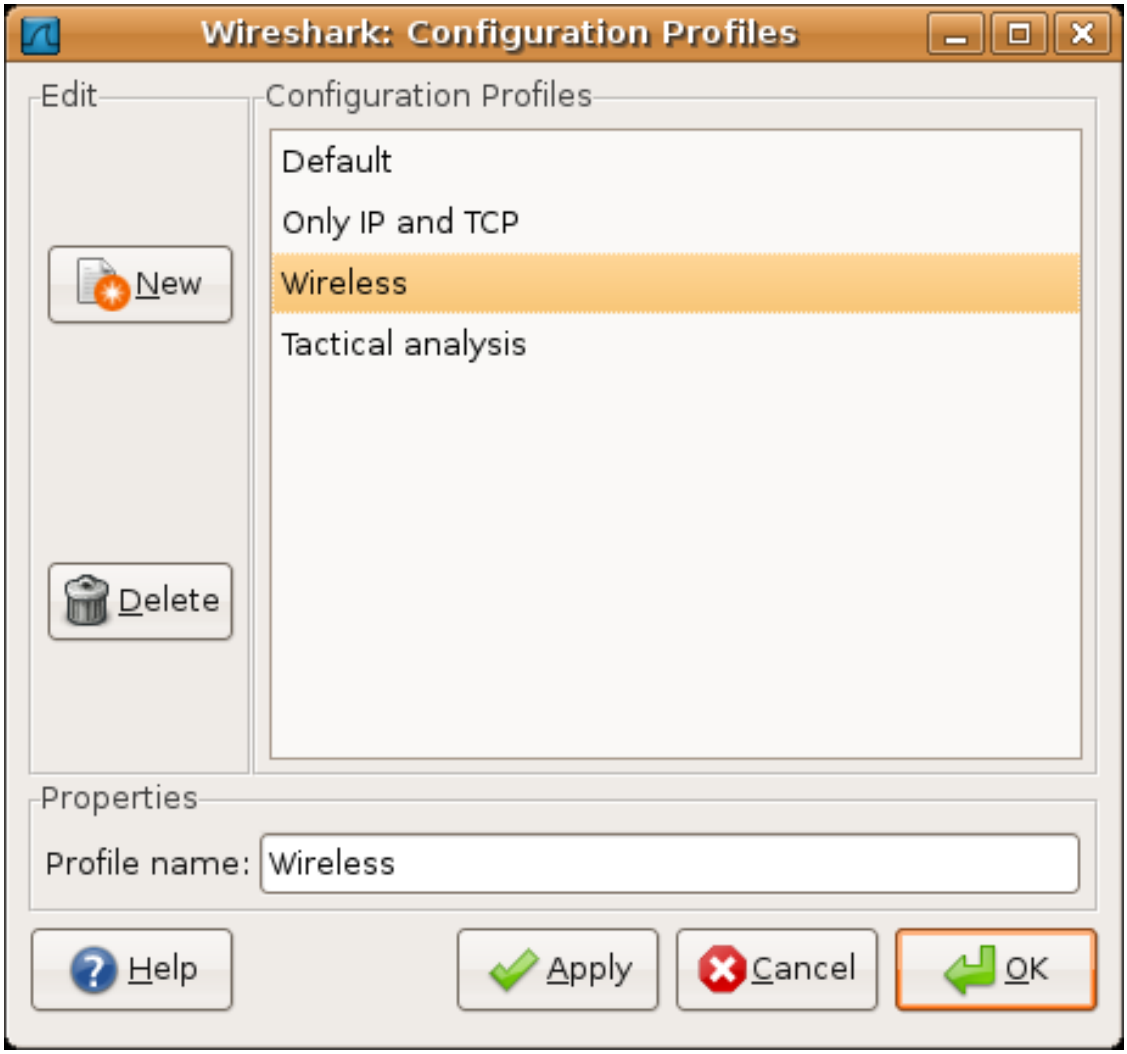
Configuration files stored in the Profiles:

- Preferences (preferences) (Section 9.5, “Preferences”)
- Capture Filters (cfilters) (Section 6.6, “Defining and saving filters”)
- Display Filters (dfilters) (Section 6.6, “Defining and saving filters”)
- Coloring Rules (colorfilters) (Section 9.3, “Packet colorization”)
- Disabled Protocols (disabled_protos) (Section 9.4.1, “The "Enabled Protocols" dialog box”)
- User Accessible Tables:
  - Display Filter Macros (dfilter_macros) (Section 9.8, “Display Filter Macros”)
  - GeoIP Database Paths (geoip_db_paths) (Section 9.9, “GeoIP Database Paths”)
  - K12 Protocols (k12_protos) (Section 9.10, “Tektronix K12xx/15 RF5 protocols Table”)
  - PRES Users Context List (pres_context_list) (Section 9.11, “PRES Users Context List”)
  - SCCP Users Table (sccp_users) (Section 9.12, “SCCP users Table”)
  - SMI Modules (smi_modules) (Section 9.13, “SMI (MIB and PIB) Modules”)
  - SMI Paths (smi_paths) (Section 9.14, “SMI (MIB and PIB) Paths”)
  - SNMP Users (snmp_users) (Section 9.15, “SNMP users Table”)
  - User DLTs Table (user_dlts) (Section 9.16, “User DLTs protocol table”)
- Some recent settings (recent), such as pane sizes in the Main window (Section 3.3, “The Main window”), column widths in the packet list (Section 3.16, “The "Packet List" pane”), all selections in the "View" menu (Section 3.7, “The "View" menu”) and the last directory navigated to in the File Open dialog.

Other configurations

All other configurations are stored in the personal configuration folder, and are common to all profiles.

Figure 9.10. The configuration profiles dialog box



New This button adds a new profile to the profiles list. The name of the created profile is "New profile" and can be changed in the Properties field.

Delete This button deletes the selected profile, including all configuration files used in this profile. It is not possible to delete the "Default" profile.

Configuration Profiles You can select a configuration profile from this list (which will fill in the profile name in the fields down at the bottom of the dialog box).

Profile name: You can change the name of the currently selected profile here.

Used as a folder name

The profile name will be used as a folder name in the configured "Personal configurations" folder. If adding multiple profiles with the same name, only one profile will be created.

Illegal characters

On Windows the profile name cannot start or end with a period (.), and cannot contain any of the following characters: \ / **:** ***** **?** " < > **|**

On Unix the profile name cannot contain the '/' character.

OK This button saves all changes, applies the selected profile and closes the dialog.

Apply This button saves all changes, applies the selected profile and keeps the dialog open.

Cancel Close this dialog. This will discard unsaved settings, new profiles will not be added and deleted profiles will not be deleted.

Help Show this help page.

9.7. User Table

The User Table editor is used for managing various tables in wireshark. Its main dialog works very similarly to that of Section 9.3, “Packet colorization”.

9.8. Display Filter Macros

Display Filter Macros are a mechanism to create shortcuts for complex filters. For example defining a display filter macro named **tcp_conv** whose text is **((ip.src == $1 and ip.dst == $2 and tcp.srcport == $3 and tcp.dstport == $4) or (ip.src == $2 and ip.dst == $1 and tcp.srcport == $4 and tcp.dstport == $3))** would allow to use a display filter like **${tcp_conv:10.1.1.2;10.1.1.3;1200;1400}** instead of typing the whole filter.

Display Filter Macros can be managed with a Section 9.7, “User Table” by selecting the **Display Filter Macros** menu item from the **View** Menu. The User Table has the following fields

Name The name of the macro.

Text The replacement text for the macro it uses $1, $2, $3, ... as the input arguments.

9.9. GeoIP Database Paths

If your copy of Wireshark supports MaxMind's GeoIP library, you can use their databases to match IP addresses to countries, cites, autonomous system numbers, ISPs, and other bits of information. Some databases are available at no cost, while others require a licensing fee. See the MaxMind web site for more information.

This table is handled by an Section 9.7, “User Table” with the following fields.

Database pathname This specifies a directory containing GeoIP data files. Any files beginning with `Geo` and ending with `.dat` will be automatically loaded. A total of 8 files can be loaded.

The locations for your data files are up to you, but `/usr/share/GeoIP` (Linux), `C:\GeoIP` (Windows), `C:\Program Files\Wireshark\GeoIP` (Windows) might be good choices.

9.10. Tektronix K12xx/15 RF5 protocols Table

The Tektronix K12xx/15 rf5 file format uses helper files (*.stk) to identify the various protocols that are used by a certain interface. Wireshark doesn't read these stk files, it uses a table that helps it identify which lowest layer protocol to use.

Stk file to protocol matching is handled by an Section 9.7, “User Table” with the following fields.

Match string A partial match for an stk filename, the first match wins, so if you have a specific case and a general one the specific one must appear first in the list.

Protocol This is the name of the encapsulating protocol (the lowest layer in the packet data) it can be either just the name of the protocol (e.g. mtp2, eth_witoutfcs, sscf-nni) or the name of the encapsulation protocol and the "application" protocol over it separated by a colon (e.g sscop:sscf-nni, sscop:alcap, sscop:nbap, ...)

9.11. PRES Users Context List

Wireshark uses this table to map a presentation context identifier to a given object identifier when the capture does not contain a PRES package with a presentation context definition list for the conversation.

This table is handled by an Section 9.7, “User Table” with the following fields.

Context Id An Integer representing the presentation context identifier for which this association is valid.

Syntax Name OID The object identifier representing the abstract syntax name, which defines the protocol that is carried over this association.

9.12. SCCP users Table

Wireshark uses this table to map specific protocols to a certain DPC/SSN combination for SCCP.

This table is handled by an Section 9.7, “User Table” with the following fields.

Network Indicator An Integer representing the network indicator for which this association is valid.

Called DPCs An range of integers representing the dpcs for which this association is valid.

Called SSNs An range of integers representing the ssns for which this association is valid.

User protocol The protocol that is carried over this association

9.13. SMI (MIB and PIB) Modules

If your copy of Wireshark supports libSMI, you can specify a list of MIB and PIB modules here. The COPS and SNMP dissectors can use them to resolve OIDs.

Module name The name of the module, e.g. IF-MIB.

9.14. SMI (MIB and PIB) Paths

If your copy of Wireshark supports libSMI, you can specify one or more paths to MIB and PIB modules here.

Directory name A module directory, e.g. `/usr/local/snmp/mibs`. Wireshark automatically uses the standard SMI path for your system, so you usually don't have to add anything here.

9.15. SNMP users Table

Wireshark uses this table to verify authentication and to decrypt encrypted SNMPv3 packets.

This table is handled by an Section 9.7, “User Table” with the following fields.

Engine ID — If given this entry will be used only for packets whose engine id is this. This field takes an hexadecimal string in the form 0102030405.

Username — This is the userName. When a single user has more than one password for different SNMP-engines the first entry to match both is taken, if you need a catch all engine-id (empty) that entry should be the last one.

Authentication model — Which auth model to use (either "MD5" or "SHA1").

Password — The authentication password. Use '\xDD' for unprintable characters. An hexadecimal password must be entered as a sequence of '\xDD' characters. For example the hex password 010203040506 must be entered as '\x01\x02\x03\x04\x05\x06'.

Privacy protocol — Which encryption algorithm to use (either "DES" or "AES").

Privacy password — The privacy password. Use '\xDD' for unprintable characters. An hexadecimal password must be entered as a sequence of '\xDD' characters. For example the hex password 010203040506 must be entered as '\x01\x02\x03\x04\x05\x06'.

9.16. User DLTs protocol table

When a pcap file uses one of the user DLTs (147 to 162) wireshark uses this table to know which protocol(s) to use for each user DLT.

This table is handled by an Section 9.7, “User Table” with the following fields.

DLT	One of the user dlts.
Payload protocol	This is the name of the payload protocol (the lowest layer in the packet data). (e.g. "eth" for ethernet, "ip" for IPv4)
Header size	If there is a header protocol (before the payload protocol) this tells which size this header is. A value of 0 disables the header protocol.
Header protocol	The name of the header protocol to be used (uses "data" as default).
Trailer size	If there is a trailer protocol (after the payload protocol) this tells which size this trailer is. A value of 0 disables the trailer protocol.
Trailer protocol	The name of the trailer protocol to be used (uses "data" as default).

9.17. IKEv2 decryption table

Wireshark can decrypt Encrypted Payloads of IKEv2 (Internet Key Exchange version 2) packets if necessary information is provided. Note that you can decrypt only IKEv2 packets with this feature. If you want to decrypt IKEv1 packets or ESP packets, use Log Filename setting under ISAKMP protocol preference or settings under ESP protocol preference respectively.

This table is handled by an Section 9.7, “User Table” with the following fields.

Initiator's SPI — Initiator's SPI of the IKE_SA. This field takes hexadecimal string without "0x" prefix and the length must be 16 hex chars (represents 8 octets).

Responder's SPI — Responder's SPI of the IKE_SA. This field takes hexadecimal string without "0x" prefix and the length must be 16 hex chars (represents 8 octets).

SK_ei — Key used to encrypt/decrypt IKEv2 packets from initiator to responder. This field takes hexadecimal string without "0x" prefix and its length must meet the requirement of the encryption algorithm selected.

SK_er — Key used to encrypt/decrypt IKEv2 packets from responder to initiator. This field takes hexadecimal string without "0x" prefix and its length must meet the requirement of the encryption algorithm selected.

Encryption Algorithm — Encryption algorithm of the IKE_SA.

SK_ai — Key used to calculate Integrity Checksum Data for IKEv2 packets from responder to initiator. This field takes hexadecimal string without "0x" prefix and its length must meet the requirement of the integrity algorithm selected.

SK_ar — Key used to calculate Integrity Checksum Data for IKEv2 packets from initiator to responder. This field takes hexadecimal string without "0x" prefix and its length must meet the requirement of the integrity algorithm selected.

Integrity Algorithm — Integrity algorithm of the IKE_SA.

Chapter 10. Lua Support in Wireshark

10.1. Introduction

Wireshark has an embedded Lua interpreter. Lua is a powerful light-weight programming language designed for extending applications. Lua is designed and implemented by a team at PUC-Rio, the Pontifical Catholic University of Rio de Janeiro in Brazil. Lua was born and raised at Tecgraf, the Computer Graphics Technology Group of PUC-Rio, and is now housed at Lua.org. Both Tecgraf and Lua.org are laboratories of the Department of Computer Science.

In Wireshark Lua can be used to write dissectors and taps.

Wireshark's Lua interpreter starts by loading **init.lua** that is located in the global configuration directory of Wireshark. Lua is disabled by default by setting the variable **disable_lua** to true in **init.lua**. To enable lua the line that sets that variable must be removed or commented out.

After loading **init.lua** from the data directory if lua is enabled Wireshark will try to load a file named **init.lua** in the user's directory.

The command line option **-X lua_script:<file.lua>** can be used to load lua scripts as well.

The Lua code will be executed once after all the protocols have being initialized and before reading any file.

10.2. Example of Dissector written in Lua

```
do
        local p_multi = Proto("multi","MultiProto");

        local vs_protos = {
                [2] = "mtp2",
                [3] = "mtp3",
                [4] = "alcap",
                [5] = "h248",
                [6] = "ranap",
                [7] = "rnsap",
                [8] = "nbap"
        }

        local f_proto = ProtoField.uint8("multi.protocol","Protocol",base.DEC,vs_protos)
        local f_dir = ProtoField.uint8("multi.direction","Direction",base.DEC,{ [1] = "incoming", [0] =
        local f_text = ProtoField.string("multi.text","Text")

        p_multi.fields = { f_proto, f_dir, f_text }

        local data_dis = Dissector.get("data")

        local protos = {
                [2] = Dissector.get("mtp2"),
                [3] = Dissector.get("mtp3"),
                [4] = Dissector.get("alcap"),
                [5] = Dissector.get("h248"),
                [6] = Dissector.get("ranap"),
                [7] = Dissector.get("rnsap"),
                [8] = Dissector.get("nbap"),
                [9] = Dissector.get("rrc"),
                [10] = DissectorTable.get("sctp.ppi"):get_dissector(3), -- m3ua
                [11] = DissectorTable.get("ip.proto"):get_dissector(132), -- sctp
        }

        function p_multi.dissector(buf,pkt,root)

                local t = root:add(p_multi,buf(0,2))
                t:add(f_proto,buf(0,1))
                t:add(f_dir,buf(1,1))

                local proto_id = buf(0,1):uint()

                local dissector = protos[proto_id]

                if dissector ~= nil then
                        dissector:call(buf(2):tvb(),pkt,root)
                elseif proto_id < 2 then
                        t:add(f_text,buf(2))
                        -- pkt.cols.info:set(buf(2,buf:len() - 3):string())
                else
                        data_dis:call(buf(2):tvb(),pkt,root)
                end

        end

        local wtap_encap_table = DissectorTable.get("wtap_encap")
        local udp_encap_table = DissectorTable.get("udp.port")

        wtap_encap_table:add(wtap.USER15,p_multi)
        wtap_encap_table:add(wtap.USER12,p_multi)
        udp_encap_table:add(7555,p_multi)
end
```

10.3. Example of Listener written in Lua

```
-- This program will register a menu that will open a window with a count of occurrences
-- of every address in the capture

do
	local function menuable_tap()
		-- Declare the window we will use
		local tw = TextWindow.new("Address Counter")

		-- This will contain a hash of counters of appearances of a certain address
		local ips = {}

		-- this is our tap
		local tap = Listener.new();

		function remove()
			-- this way we remove the listener than otherwise will remain running indifinit
			tap:remove();
		end

		-- we tell the window to call the remove() function when closed
		tw:set_atclose(remove)

		-- this function will be called once for each packet
		function tap.packet(pinfo,tvb)
			local src = ips[tostring(pinfo.src)] or 0
			local dst = ips[tostring(pinfo.dst)] or 0

			ips[tostring(pinfo.src)] = src + 1
			ips[tostring(pinfo.dst)] = dst + 1
		end

		-- this function will be called once every few seconds to update our window
		function tap.draw(t)
			tw:clear()
			for ip,num in pairs(ips) do
				tw:append(ip .. "\t" .. num .. "\n");
			end
		end

		-- this function will be called whenever a reset is needed
		-- e.g. when reloading the capture file
		function tap.reset()
			tw:clear()
			ips = {}
		end
	end

	-- using this function we register our fuction
	-- to be called when the user selects the Tools->Test->Packets menu
	register_menu("Test/Packets", menuable_tap, MENU_TOOLS)
end
```

10.4. Wireshark's Lua API Reference Manual

This Part of the User Guide describes the Wireshark specific functions in the embedded Lua.

10.4.1. Saving capture files

10.4.1.1. Dumper

10.4.1.1.1. Dumper.new(filename, [filetype], [encap])

Creates a file to write packets. Dumper:new_for_current() will probably be a better choice.

10.4.1.1.1.1. Arguments

filename	The name of the capture file to be created
filetype (optional)	The type of the file to be created
encap (optional)	The encapsulation to be used in the file to be created

10.4.1.1.1.2. Returns

The newly created Dumper object

10.4.1.1.1.3. Errors

- Not every filetype handles every encap

10.4.1.1.2. dumper:close()

Closes a dumper

10.4.1.1.2.1. Errors

- Cannot operate on a closed dumper

10.4.1.1.3. dumper:flush()

Writes all unsaved data of a dumper to the disk.

10.4.1.1.4. dumper:dump(timestamp, pseudoheader, bytearray)

Dumps an arbitrary packet. Note: Dumper:dump_current() will fit best in most cases.

10.4.1.1.4.1. Arguments

timestamp	The absolute timestamp the packet will have
pseudoheader	The Pseudoheader to use.

bytearray the data to be saved

10.4.1.1.5. dumper:new_for_current([filetype])

Creates a capture file using the same encapsulation as the one of the cuurrent packet

10.4.1.1.5.1. Arguments

filetype (optional) The file type. Defaults to pcap.

10.4.1.1.5.2. Returns

The newly created Dumper Object

10.4.1.1.5.3. Errors

- Cannot be used outside a tap or a dissector

10.4.1.1.6. dumper:dump_current()

Dumps the current packet as it is.

10.4.1.1.6.1. Errors

- Cannot be used outside a tap or a dissector

10.4.1.2. PseudoHeader

A pseudoheader to be used to save captured frames.

10.4.1.2.1. PseudoHeader.none()

Creates a "no" pseudoheader.

10.4.1.2.1.1. Returns

A null pseudoheader

10.4.1.2.2. PseudoHeader.eth([fcslen])

Creates an ethernet pseudoheader

10.4.1.2.2.1. Arguments

fcslen (optional) The fcs length

10.4.1.2.2.2. Returns

The ethernet pseudoheader

10.4.1.2.3. PseudoHeader.atm([aal], [vpi], [vci], [channel], [cells], [aal5u2u], [aal5len])

Creates an ATM pseudoheader

10.4.1.2.3.1. Arguments

aal (optional)	AAL number
vpi (optional)	VPI
vci (optional)	VCI
channel (optional)	Channel
cells (optional)	Number of cells in the PDU
aal5u2u (optional)	AAL5 User to User indicator
aal5len (optional)	AAL5 Len

10.4.1.2.3.2. Returns

The ATM pseudoheader

10.4.1.2.4. PseudoHeader.mtp2()

Creates an MTP2 PseudoHeader

10.4.1.2.4.1. Returns

The MTP2 pseudoheader

10.4.2. Obtaining dissection data

10.4.2.1. Field

A Field extractor to to obtain field values.

10.4.2.1.1. Field.new(fieldname)

Create a Field extractor

10.4.2.1.1.1. Arguments

fieldname	The filter name of the field (e.g. ip.addr)

10.4.2.1.1.2. Returns

The field extractor

10.4.2.1.1.3. Errors

- A Field extractor must be defined before Taps or Dissectors get called

10.4.2.1.2. field:__call()

Obtain all values (see FieldInfo) for this field.

10.4.2.1.2.1. Returns

All the values of this field

10.4.2.1.2.2. Errors

- Fields cannot be used outside dissectors or taps

10.4.2.2. FieldInfo

An extracted Field

10.4.2.2.1. fieldinfo:__len()

Obtain the Length of the field

10.4.2.2.2. fieldinfo:__unm()

Obtain the Offset of the field

10.4.2.2.3. fieldinfo:__call()

Obtain the Value of the field

10.4.2.2.4. fieldinfo:__tostring()

The string representation of the field

10.4.2.2.5. fieldinfo:__eq()

Checks whether lhs is within rhs

10.4.2.2.5.1. Errors

- Data source must be the same for both fields

10.4.2.2.6. fieldinfo:__le()

Checks whether the end byte of lhs is before the end of rhs

10.4.2.2.7. fieldinfo:__lt()

Checks whether the end byte of rhs is before the beginning of rhs

10.4.2.2.7.1. Errors

- Data source must be the same for both fields

10.4.2.2.8. fieldinfo.name

The name of this field

10.4.2.2.9. fieldinfo.label

The string representing this field

10.4.2.2.10. fieldinfo.value

The value of this field

10.4.2.2.11. fieldinfo.len

The length of this field

10.4.2.2.12. fieldinfo.offset

The offset of this field

10.4.2.3. Non Method Functions

10.4.2.3.1. all_field_infos()

Obtain all fields from the current tree

10.4.2.3.1.1. Errors

- Cannot be called outside a listener or dissector

10.4.3. GUI support

10.4.3.1. ProgDlg

Manages a progress bar dialog.

10.4.3.1.1. ProgDlg.new([title], [task])

Creates a new TextWindow.

10.4.3.1.1.1. Arguments

title (optional)	Title of the new window, defaults to "Progress".
task (optional)	Current task, defaults to "".

10.4.3.1.1.2. Returns

The newly created TextWindow object.

10.4.3.1.2. progdlg:update(progress, [task])

Appends text

10.4.3.1.2.1. Arguments

progress	Part done (e.g. 0.75).
task (optional)	Current task, defaults to "".

10.4.3.1.2.2. Errors

- Cannot be called for something not a ProgDlg
- Progress value out of range (must be between 0.0 and 1.0)

10.4.3.1.3. progdlg:stopped()

Checks wheher the user has pressed the stop button.

10.4.3.1.3.1. Returns

true if the user has asked to stop the progress.

10.4.3.1.3.2. Errors

- Cannot be called for something not a ProgDlg

10.4.3.1.4. progdlg:close()

Appends text

10.4.3.1.4.1. Errors

- Cannot be called for something not a ProgDlg

10.4.3.2. TextWindow

Manages a text window.

10.4.3.2.1. TextWindow.new([title])

Creates a new TextWindow.

10.4.3.2.1.1. Arguments

title (optional)	Title of the new window.

10.4.3.2.1.2. Returns

The newly created TextWindow object.

10.4.3.2.2. textwindow:set_atclose(action)

Set the function that will be called when the window closes

10.4.3.2.2.1. Arguments

action A function to be executed when the user closes the window

10.4.3.2.2.2. Returns

The TextWindow object.

10.4.3.2.2.3. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.3. textwindow:set(text)

Sets the text.

10.4.3.2.3.1. Arguments

text The text to be used.

10.4.3.2.3.2. Returns

The TextWindow object.

10.4.3.2.3.3. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.4. textwindow:append(text)

Appends text

10.4.3.2.4.1. Arguments

text The text to be appended

10.4.3.2.4.2. Returns

The TextWindow object.

10.4.3.2.4.3. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.5. textwindow:prepend(text)

Prepends text

10.4.3.2.5.1. Arguments

text The text to be appended

10.4.3.2.5.2. Returns

The TextWindow object.

10.4.3.2.5.3. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.6. textwindow:clear()

Erases all text in the window.

10.4.3.2.6.1. Returns

The TextWindow object.

10.4.3.2.6.2. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.7. textwindow:get_text()

Get the text of the window

10.4.3.2.7.1. Returns

The TextWindow's text.

10.4.3.2.7.2. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.8. textwindow:set_editable([editable])

Make this window editable

10.4.3.2.8.1. Arguments

editable (optional) A boolean flag, defaults to true

10.4.3.2.8.2. Returns

The TextWindow object.

10.4.3.2.8.3. Errors

- Cannot be called for something not a TextWindow

10.4.3.2.9. textwindow:add_button(label, function)

10.4.3.2.9.1. Arguments

label	The label of the button
function	The function to be called when clicked

10.4.3.2.9.2. Returns

The TextWindow object.

10.4.3.2.9.3. Errors

- Cannot be called for something not a TextWindow

10.4.3.3. Non Method Functions

10.4.3.3.1. gui_enabled()

Checks whether the GUI facility is enabled.

10.4.3.3.1.1. Returns

A boolean: true if it is enabled, false if it isn't.

10.4.3.3.2. register_menu(name, action, [group])

Register a menu item in one of the main menus.

10.4.3.3.2.1. Arguments

name	The name of the menu item. The submenus are to be separated by '/'s. (string)
action	The function to be called when the menu item is invoked. (function taking no arguments and returning nothing)
group (optional)	The menu group into which the menu item is to be inserted. If omitted, defaults to MENU_STAT_GENERIC. One of MENU_STAT (Statistics), MENU_STAT_GENERIC (Statistics, first section), MENU_STAT_CONVERSATION (Statistics/Conversation List), MENU_STAT_ENDPOINT (Statistics/Endpoint List), MENU_STAT_RESPONSE (Statistics/Service Response Time), MENU_STAT_TELEPHONY (Telephony), MENU_ANALYZE (Analyze), MENU_ANALYZE_CONVERSATION (Analyze/Conversation Filter), MENU_TOOLS (Tools). (number)

10.4.3.3.3. new_dialog(title, action, ...)

Pops up a new dialog

10.4.3.3.3.1. Arguments

title	Title of the dialog's window.
action	Action to be performed when OKd.
...	A series of strings to be used as labels of the dialog's fields

10.4.3.3.3.2. Errors

- At least one field required
- All fields must be strings

10.4.3.3.4. retap_packets()

Rescan all packets and just run taps - don't reconstruct the display.

10.4.3.3.5. copy_to_clipboard(text)

Copy a string into the clipboard

10.4.3.3.5.1. Arguments

text	The string to be copied into the clipboard.

10.4.3.3.6. open_capture_file(filename, filter)

Open and display a capture file

10.4.3.3.6.1. Arguments

filename	The name of the file to be opened.
filter	A filter to be applied as the file gets opened.

10.4.3.3.7. set_filter(text)

Set the main filter text

10.4.3.3.7.1. Arguments

text	The filter's text.

10.4.3.3.8. apply_filter()

Apply the filter in the main filter box

10.4.3.3.9. reload()

Reload the current capture file

10.4.3.3.10. browser_open_url(url)

Open an url in a browser

10.4.3.3.10.1. Arguments

url The url.

10.4.3.3.11. browser_open_data_file(filename)

Open an file in a browser

10.4.3.3.11.1. Arguments

filename The url.

10.4.4. Post-dissection packet analysis

10.4.4.1. Listener

A Listener, is called once for every packet that matches a certain filter or has a certain tap. It can read the tree, the packet's Tvb eventually the tapped data but it cannot add elements to the tree.

10.4.4.1.1. Listener.new([tap], [filter])

Creates a new Listener listener

10.4.4.1.1.1. Arguments

tap (optional)	The name of this tap
filter (optional)	A filter that when matches the tap.packet function gets called (use nil to be called for every packet)

10.4.4.1.1.2. Returns

The newly created Listener listener object

10.4.4.1.1.3. Errors

- tap registration error

10.4.4.1.2. listener:remove()

Removes a tap listener

10.4.4.1.3. listener.packet

A function that will be called once every packet matches the Listener listener filter. function tap.packet(pinfo,tvb,userdata) ... end

10.4.4.1.4. listener.draw

A function that will be called once every few seconds to redraw the gui objects in tshark this funtion is called oly at the very end of the capture file. function tap.draw(userdata) ... end

10.4.4.1.5. listener.reset

A function that will be called at the end of the capture run. function tap.reset(userdata) ... end

10.4.5. Obtaining packet information

10.4.5.1. Address

Represents an address

10.4.5.1.1. Address.ip(hostname)

Creates an Address Object representing an IP address.

10.4.5.1.1.1. Arguments

hostname The address or name of the IP host.

10.4.5.1.1.2. Returns

The Address object

10.4.5.1.2. address:__tostring()

10.4.5.1.2.1. Returns

The string representing the address.

10.4.5.1.3. address:__eq()

Compares two Addresses

10.4.5.1.4. address:__le()

Compares two Addresses

10.4.5.1.5. address:__lt()

Compares two Addresses

10.4.5.2. Column

A Column in the packet list

10.4.5.2.1. column:__tostring()

10.4.5.2.1.1. Returns

A string representing the column

10.4.5.2.2. column:clear()

Clears a Column

10.4.5.2.3. column:set(text)

Sets the text of a Column

10.4.5.2.3.1. Arguments

text The text to which to set the Column

10.4.5.2.4. column:append(text)

Appends text to a Column

10.4.5.2.4.1. Arguments

text The text to append to the Column

10.4.5.2.5. column:preppend(text)

Prepends text to a Column

10.4.5.2.5.1. Arguments

text The text to prepend to the Column

10.4.5.3. Columns

The Columns of the packet list.

10.4.5.3.1. columns:__tostring()

10.4.5.3.1.1. Returns

The string "Columns", no real use, just for debugging purposes.

10.4.5.3.2. columns:__newindex(column, text)

Sets the text of a specific column

10.4.5.3.2.1. Arguments

column The name of the column to set

text The text for the column

10.4.5.4. Pinfo

Packet information

10.4.5.4.1. pinfo.number

The number of this packet in the current file

10.4.5.4.2. pinfo.len

The length of the frame

10.4.5.4.3. pinfo.caplen

The captured length of the frame

10.4.5.4.4. pinfo.abs_ts

When the packet was captured

10.4.5.4.5. pinfo.rel_ts

Number of seconds passed since beginning of capture

10.4.5.4.6. pinfo.delta_ts

Number of seconds passed since the last captured packet

10.4.5.4.7. pinfo.delta_dis_ts

Number of seconds passed since the last displayed packet

10.4.5.4.8. pinfo.visited

Whether this packet hass been already visited

10.4.5.4.9. pinfo.src

Source Address of this Packet

10.4.5.4.10. pinfo.dst

Destination Address of this Packet

10.4.5.4.11. pinfo.lo

lower Address of this Packet

10.4.5.4.12. pinfo.hi

higher Address of this Packet

10.4.5.4.13. pinfo.dl_src

Data Link Source Address of this Packet

10.4.5.4.14. pinfo.dl_dst

Data Link Destination Address of this Packet

10.4.5.4.15. pinfo.net_src

Network Layer Source Address of this Packet

10.4.5.4.16. pinfo.net_dst

Network Layer Destination Address of this Packet

10.4.5.4.17. pinfo.ptype

Type of Port of .src_port and .dst_port

10.4.5.4.18. pinfo.src_port

Source Port of this Packet

10.4.5.4.19. pinfo.dst_port

Source Address of this Packet

10.4.5.4.20. pinfo.ipproto

IP Protocol id

10.4.5.4.21. pinfo.circuit_id

For circuit based protocols

10.4.5.4.22. pinfo.match

Port/Data we are matching

10.4.5.4.23. pinfo.curr_proto

Which Protocol are we dissecting

10.4.5.4.24. pinfo.columns

Accesss to the packet list columns

10.4.5.4.25. pinfo.cols

Accesss to the packet list columns (equivalent to pinfo.columns)

10.4.5.4.26. pinfo.desegment_len

Estimated number of additional bytes required for completing the PDU

10.4.5.4.27. pinfo.desegment_offset

Offset in the tvbuff at which the dissector will continue processing when next called

10.4.5.4.28. pinfo.private_data

Access to private data

10.4.6. Functions for writing dissectors

10.4.6.1. Dissector

A refererence to a dissector, used to call a dissector against a packet or a part of it.

10.4.6.1.1. Dissector.get(name)

Obtains a dissector reference by name

10.4.6.1.1.1. Arguments

name	The name of the dissector

10.4.6.1.1.2. Returns

The Dissector reference

10.4.6.1.2. dissector:call(tvb, pinfo, tree)

Calls a dissector against a given packet (or part of it)

10.4.6.1.2.1. Arguments

tvb	The buffer to dissect
pinfo	The packet info
tree	The tree on which to add the protocol items

10.4.6.2. DissectorTable

A table of subdissectors of a particular protocol (e.g. TCP subdissectors like http, smtp, sip are added to table "tcp.port"). Useful to add more dissectors to a table so that they appear in the Decode As... dialog.

10.4.6.2.1. DissectorTable.new(tablename, [uiname], [type], [base])

Creates a new DissectorTable for your dissector's use.

10.4.6.2.1.1. Arguments

tablename	The short name of the table.
uiname (optional)	The name of the table in the User Interface (defaults to the name given).
type (optional)	Either FT_UINT* or FT_STRING (defaults to FT_UINT32)

base (optional)	Either BASE_NONE, BASE_DEC, BASE_HEX, BASE_OCT, BASE_DEC_HEX or BASE_HEX_DEC (defaults to BASE_DEC)

10.4.6.2.1.2. Returns

The newly created DissectorTable

10.4.6.2.2. DissectorTable.get(tablename)

Obtain a reference to an existing dissector table.

10.4.6.2.2.1. Arguments

tablename	The short name of the table.

10.4.6.2.2.2. Returns

The DissectorTable

10.4.6.2.3. dissectortable:add(pattern, dissector)

Add a dissector to a table.

10.4.6.2.3.1. Arguments

pattern	The pattern to match (either an integer or a string depending on the table's type).
dissector	The dissector to add (either an Proto or a Dissector).

10.4.6.2.4. dissectortable:remove(pattern, dissector)

Remove a dissector from a table

10.4.6.2.4.1. Arguments

pattern	The pattern to match (either an integer or a string depending on the table's type).
dissector	The dissector to add (either an Proto or a Dissector).

10.4.6.2.5. dissectortable:try(pattern, tvb, pinfo, tree)

Try to call a dissector from a table

10.4.6.2.5.1. Arguments

pattern	The pattern to be matched (either an integer or a string depending on the table's type).
tvb	The buffer to dissect
pinfo	The packet info

tree The tree on which to add the protocol items

10.4.6.2.6. dissectortable:get_dissector(pattern)

Try to obtain a dissector from a table.

10.4.6.2.6.1. Arguments

pattern The pattern to be matched (either an integer or a string depending on the table's type).

10.4.6.2.6.2. Returns

The dissector handle if found

nil if not found

10.4.6.3. Pref

A preference of a Protocol.

10.4.6.3.1. Pref.bool(label, default, descr)

Creates a boolean preference to be added to a Protocol's prefs table.

10.4.6.3.1.1. Arguments

label The Label (text in the right side of the preference input) for this preference

default The default value for this preference

descr A description of what this preference is

10.4.6.3.2. Pref.uint(label, default, descr)

Creates an (unsigned) integer preference to be added to a Protocol's prefs table.

10.4.6.3.2.1. Arguments

label The Label (text in the right side of the preference input) for this preference

default The default value for this preference

descr A description of what this preference is

10.4.6.3.3. Pref.string(label, default, descr)

Creates a string preference to be added to a Protocol's prefs table.

10.4.6.3.3.1. Arguments

label The Label (text in the right side of the preference input) for this preference

default	The default value for this preference
descr	A description of what this preference is

10.4.6.3.4. Pref.enum(label, default, descr, enum, radio)

Creates an enum preference to be added to a Protocol's prefs table.

10.4.6.3.4.1. Arguments

label	The Label (text in the right side of the preference input) for this preference
default	The default value for this preference
descr	A description of what this preference is
enum	A enum table
radio	Radio button (true) or Combobox (false)

10.4.6.3.5. Pref.range(label, default, descr, range, max)

Creates a range preference to be added to a Protocol's prefs table.

10.4.6.3.5.1. Arguments

label	The Label (text in the right side of the preference input) for this preference
default	The default value for this preference
descr	A description of what this preference is
range	The range
max	The maximum value

10.4.6.3.6. Pref.statictext(label, descr)

Creates a static text preference to be added to a Protocol's prefs table.

10.4.6.3.6.1. Arguments

label	The static text
descr	The static text description

10.4.6.4. Prefs

The table of preferences of a protocol

10.4.6.4.1. prefs:__newindex(name, pref)

Creates a new preference

10.4.6.4.1.1. Arguments

name The abbreviation of this preference

pref A valid but still unassigned Pref object

10.4.6.4.1.2. Errors

- Unknow Pref type

10.4.6.4.2. prefs:__index(name)

Get the value of a preference setting

10.4.6.4.2.1. Arguments

name The abbreviation of this preference

10.4.6.4.2.2. Returns

The current value of the preference

10.4.6.4.2.3. Errors

- Unknow Pref type

10.4.6.5. Proto

A new protocol in wireshark. Protocols have more uses, the main one is to dissect a protocol. But they can be just dummies used to register preferences for other purposes.

10.4.6.5.1. Proto.new(name, desc)

10.4.6.5.1.1. Arguments

name The name of the protocol

desc A Long Text description of the protocol (usually lowercase)

10.4.6.5.1.2. Returns

The newly created protocol

10.4.6.5.2. proto.dissector

The protocol's dissector, a function you define

10.4.6.5.3. proto.fields

The Fields Table of this dissector

10.4.6.5.4. proto.prefs

The preferences of this dissector

10.4.6.5.5. proto.init

The init routine of this dissector, a function you define

10.4.6.5.6. proto.name

The name given to this dissector

10.4.6.6. ProtoField

A Protocol field (to be used when adding items to the dissection tree)

10.4.6.6.1. ProtoField.new(name, abbr, type, [valuestring], [base], [mask], [descr])

Creates a new field to be used in a protocol.

10.4.6.6.1.1. Arguments

name	Actual name of the field (the string that appears in the tree).
abbr	Filter name of the field (the string that is used in filters).
type	Field Type (FT_*).
valuestring (optional)	A ValueString object.
base (optional)	The representation BASE_*.
mask (optional)	The bitmask to be used.
descr (optional)	The description of the field.

10.4.6.6.1.2. Returns

The newly created ProtoField object

10.4.6.6.2. ProtoField.uint8(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.2.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values

mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.2.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.3. ProtoField.uint16(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.3.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.3.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.4. ProtoField.uint24(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.4.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.4.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.5. ProtoField.uint32(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.5.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)

name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.5.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.6. ProtoField.uint64(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.6.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.6.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.7. ProtoField.int8(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.7.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.7.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.8. ProtoField.int16(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.8.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.8.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.9. ProtoField.int24(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.9.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.9.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.10. ProtoField.int32(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.10.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.10.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.11. ProtoField.int64(abbr, [name], [base], [valuestring], [mask], [desc])

10.4.6.6.11.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.11.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.12. ProtoField.framenum(abbr, [name], [base], [valuestring], [mask], [desc])

A frame number (for hyperlinks between frames)

10.4.6.6.12.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
base (optional)	One of base.DEC, base.HEX or base.OCT
valuestring (optional)	A table containing the text that corresponds to the values
mask (optional)	Integer mask of this field
desc (optional)	Description of the field

10.4.6.6.12.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.13. ProtoField.ipv4(abbr, [name], [desc])

10.4.6.6.13.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)

desc (optional)	Description of the field

10.4.6.6.13.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.14. ProtoField.ipv6(abbr, [name], [desc])

10.4.6.6.14.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.14.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.15. ProtoField.ether(abbr, [name], [desc])

10.4.6.6.15.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.15.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.16. ProtoField.float(abbr, [name], [desc])

10.4.6.6.16.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.16.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.17. ProtoField.double(abbr, [name], [desc])

10.4.6.6.17.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.17.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.18. ProtoField.string(abbr, [name], [desc])

10.4.6.6.18.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.18.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.19. ProtoField.stringz(abbr, [name], [desc])

10.4.6.6.19.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.19.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.20. ProtoField.bytes(abbr, [name], [desc])

10.4.6.6.20.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.20.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.21. ProtoField.ubytes(abbr, [name], [desc])

10.4.6.6.21.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.21.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.22. ProtoField.guid(abbr, [name], [desc])

10.4.6.6.22.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.22.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.23. ProtoField.oid(abbr, [name], [desc])

10.4.6.6.23.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.23.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.6.24. ProtoField.bool(abbr, [name], [desc])

10.4.6.6.24.1. Arguments

abbr	Abbreviated name of the field (the string used in filters)
name (optional)	Actual name of the field (the string that appears in the tree)
desc (optional)	Description of the field

10.4.6.6.24.2. Returns

A protofield item to be added to a ProtoFieldArray

10.4.6.7. Non Method Functions

10.4.6.7.1. register_postdissector(proto)

Make a protocol (with a dissector) a postdissector. It will be called for every frame after dissection

10.4.6.7.1.1. Arguments

proto the protocol to be used as postdissector

10.4.7. Adding information to the dissection tree

10.4.7.1. TreeItem

TreeItems represent information in the packet-details pane. A root TreeItem is passed to dissectors as first argument.

10.4.7.1.1. treeitem:add()

Adds an child item to a given item, returning the child. tree_item:add([proto_field | proto], [tvbrange], [label], ...) if the proto_field represents a numeric value (int, uint or float) is to be treated as a Big Endian (network order) Value.

10.4.7.1.1.1. Returns

The child item

10.4.7.1.2. treeitem:add_le()

Adds (and returns) an child item to a given item, returning the child. tree_item:add([proto_field | proto], [tvbrange], [label], ...) if the proto_field represents a numeric value (int, uint or float) is to be treated as a Little Endian Value.

10.4.7.1.2.1. Returns

The child item

10.4.7.1.3. treeitem:set_text(text)

Sets the text of the label

10.4.7.1.3.1. Arguments

text The text to be used.

10.4.7.1.4. treeitem:append_text(text)

Appends text to the label

10.4.7.1.4.1. Arguments

text The text to be appended.

10.4.7.1.5. treeitem:set_expert_flags([group], [severity])

Sets the expert flags of the item.

10.4.7.1.5.1. Arguments

group (optional)	One of PI_CHECKSUM, PI_SEQUENCE, PI_RESPONSE_CODE, PI_REQUEST_CODE, PI_UNDECODED, PI_REASSEMBLE, PI_MALFORMED or PI_DEBUG
severity (optional)	One of PI_CHAT, PI_NOTE, PI_WARN, PI_ERROR

10.4.7.1.6. treeitem:add_expert_info([group], [severity], [text])

Sets the expert flags of the item and adds expert info to the packet.

10.4.7.1.6.1. Arguments

group (optional)	One of PI_CHECKSUM, PI_SEQUENCE, PI_RESPONSE_CODE, PI_REQUEST_CODE, PI_UNDECODED, PI_REASSEMBLE, PI_MALFORMED or PI_DEBUG
severity (optional)	One of PI_CHAT, PI_NOTE, PI_WARN, PI_ERROR
text (optional)	The text for the expert info

10.4.7.1.7. treeitem:set_generated()

Marks the TreeItem as a generated field (with data infered but not contained in the packet).

10.4.7.1.8. treeitem:set_hidden()

Should not be used

10.4.8. Functions for handling packet data

10.4.8.1. ByteArray

10.4.8.1.1. ByteArray.new([hexbytes])

Creates a ByteArray Object

10.4.8.1.1.1. Arguments

hexbytes (optional)	A string consisting of hexadecimal bytes like "00 B1 A2" or "1a2b3c4d"

10.4.8.1.1.2. Returns

The new ByteArray object.

10.4.8.1.2. bytearray:__concat(first, second)

Concatenate two ByteArrays

10.4.8.1.2.1. Arguments

first	First array
second	Second array

10.4.8.1.2.2. Returns

The new composite ByteArray.

10.4.8.1.2.3. Errors

- Both arguments must be ByteArrays

10.4.8.1.3. bytearray:prepend(prepended)

Prepend a ByteArray to this ByteArray

10.4.8.1.3.1. Arguments

prepended	Array to be prepended

10.4.8.1.3.2. Errors

- Both arguments must be ByteArrays

10.4.8.1.4. bytearray:append(appended)

Append a ByteArray to this ByteArray

10.4.8.1.4.1. Arguments

appended	Array to be appended

10.4.8.1.4.2. Errors

- Both arguments must be ByteArrays

10.4.8.1.5. bytearray:set_size(size)

Sets the size of a ByteArray, either truncating it or filling it with zeros.

10.4.8.1.5.1. Arguments

size New size of the array

10.4.8.1.5.2. Errors

- ByteArray size must be non-negative

10.4.8.1.6. bytearray:set_index(index, value)

Sets the value of an index of a ByteArray.

10.4.8.1.6.1. Arguments

index The position of the byte to be set

value The char value to set [0-255]

10.4.8.1.7. bytearray:get_index(index)

Set the value of a byte in a ByteArray

10.4.8.1.7.1. Arguments

index The position of the byte to be set

10.4.8.1.7.2. Returns

The value [0-255] of the byte.

10.4.8.1.8. bytearray:len()

Obtain the length of a ByteArray

10.4.8.1.8.1. Returns

The length of the ByteArray.

10.4.8.1.9. bytearray:subset(offset, length)

Obtain a segment of a ByteArray

10.4.8.1.9.1. Arguments

offset The position of the first byte

length The length of the segment

10.4.8.1.9.2. Returns

A ByteArray contaning the requested segment.

A string contaning a representaion of the ByteArray.

10.4.8.2. Int

Int64 represents a 64 bit integer. Lua uses one single number representation which can be chosen at compile time and since it is often set to IEEE 754 double precision floating point, we cannot store a 64 bit integer with full precision. For details, see: http://lua-users.org/wiki/FloatingPoint

10.4.8.3. Tvb

A Tvb represents the packet's buffer. It is passed as an argument to listeners and dissectors, and can be used to extract information (via TvbRange) from the packet's data. Beware that Tvbs are usable only by the current listener or dissector call and are destroyed as soon as the listener/dissector returns, so references to them are unusable once the function has returned. To create a tvbrange the tvb must be called with offset and length as optional arguments (the offset defaults to 0 and the length to tvb:len())

10.4.8.3.1. Tvb.new_real(bytearray, name)

Creates a new Tvb from a bytearray (it gets added to the current frame too)

10.4.8.3.1.1. Arguments

bytearray	The data source for this Tvb.
name	The name to be given to the new data-source.

10.4.8.3.1.2. Returns

The created Tvb.

10.4.8.3.2. Tvb.tvb(range)

Creates a (sub)Tvb from using a TvbRange

10.4.8.3.2.1. Arguments

range	The TvbRange from which to create the new Tvb.

10.4.8.3.3. tvb:__tostring()

Convert the bytes of a Tvb into a string, to be used for debugging purposes as '...' will be appended in case the string is too long.

10.4.8.3.3.1. Returns

The string.

10.4.8.3.4. tvb:len()

Obtain the length of a TVB

10.4.8.3.4.1. Returns

The length of the Tvb.

10.4.8.3.5. tvb:offset()

Returns the raw offset (from the beginning of the source Tvb) of a sub Tvb.

10.4.8.3.5.1. Returns

The raw offset of the Tvb.

10.4.8.3.6. tvb:__call()

Equivalent to tvb:range(...)

10.4.8.4. TvbRange

A TvbRange represents an usable range of a Tvb and is used to extract data from the Tvb that generated it TvbRanges are created by calling a tvb (e.g. tvb(offset,length)). If the TvbRange span is outside the Tvb's range the creation will cause a runtime error.

10.4.8.4.1. tvb:range([offset], [length])

Creates a tvbr from this Tvb. This is used also as the Tvb:__call() metamethod.

10.4.8.4.1.1. Arguments

offset (optional)	The offset (in octets) from the begining of the Tvb. Defaults to 0.
length (optional)	The length (in octets) of the range. Defaults to until the end of the Tvb.

10.4.8.4.1.2. Returns

The TvbRange

10.4.8.4.2. tvbrange:uint()

Get a Big Endian (network order) unsigned integer from a TvbRange. The range must be 1, 2, 3 or 4 octets long.

10.4.8.4.2.1. Returns

The unsigned integer value

10.4.8.4.3. tvbrange:le_uint()

Get a Little Endian unsigned integer from a TvbRange. The range must be 1, 2, 3 or 4 octets long.

10.4.8.4.3.1. Returns

The unsigned integer value

10.4.8.4.4. tvbrange:uint64()

Get a Big Endian (network order) unsigned 64 bit integer from a TvbRange. The range must be 1-8 oct-

ets long.

10.4.8.4.5. tvbrange:le_uint64()

Get a Little Endian unsigned 64 bit integer from a TvbRange. The range must be 1-8 octets long.

10.4.8.4.6. tvbrange:float()

Get a Big Endian (network order) floating point number from a TvbRange. The range must be 4 or 8 octets long.

10.4.8.4.6.1. Returns

The flaoting point value

10.4.8.4.7. tvbrange:le_float()

Get a Little Endian floating point number from a TvbRange. The range must be 4 or 8 octets long.

10.4.8.4.7.1. Returns

The flaoting point value

10.4.8.4.8. tvbrange:ipv4()

Get an IPv4 Address from a TvbRange.

10.4.8.4.8.1. Returns

The IPv4 Address

10.4.8.4.9. tvbrange:le_ipv4()

Get an Little Endian IPv4 Address from a TvbRange.

10.4.8.4.9.1. Returns

The IPv4 Address

10.4.8.4.10. tvbrange:ether()

Get an Ethernet Address from a TvbRange.

10.4.8.4.10.1. Returns

The Ethernet Address

10.4.8.4.10.2. Errors

- The range must be 6 bytes long

10.4.8.4.11. tvbrange:string()

Obtain a string from a TvbRange

10.4.8.4.11.1. Returns

The string

10.4.8.4.12. tvbrange:bytes()

Obtain a ByteArray

10.4.8.4.12.1. Returns

The ByteArray

10.4.8.4.13. tvbrange:len()

Obtain the length of a TvbRange

10.4.8.4.14. tvbrange:offset()

Obtain the offset in a TvbRange

10.4.8.4.15. tvbrange:__tostring()

Converts the TvbRange into a string. As the string gets truncated you should use this only for debugging purposes or if what you want is to have a truncated string in the format 67:89:AB:...

10.4.8.5. UInt

Int64 represents a 64 bit integer.

10.4.9. Utility Functions

10.4.9.1. Dir

A Directory

10.4.9.1.1. Dir.open(pathname, [extension])

Usage: for filename in Dir.open(path) do ... end

10.4.9.1.1.1. Arguments

pathname	The pathname of the directory
extension (optional)	If given, only file with this extension will be returned

10.4.9.1.1.2. Returns

the Dir object

10.4.9.1.2. dir:__call()

At every invocation will return one file (nil when done)

10.4.9.1.3. dir:close()

Closes the directory

10.4.9.2. Non Method Functions

10.4.9.2.1. format_date(timestamp)

Formats an absolute timestamp into a human readable date

10.4.9.2.1.1. Arguments

timestamp A timestamp value to convert.

10.4.9.2.1.2. Returns

A string with the formated date

10.4.9.2.2. format_time(timestamp)

Formats a relative timestamp in a human readable form

10.4.9.2.2.1. Arguments

timestamp A timestamp value to convert

10.4.9.2.2.2. Returns

A string with the formated time

10.4.9.2.3. report_failure(text)

Reports a failure to the user

10.4.9.2.3.1. Arguments

text Message

10.4.9.2.4. critical(...)

Will add a log entry with critical severity

10.4.9.2.4.1. Arguments

... objects to be printed

10.4.9.2.5. warn(...)

Will add a log entry with warn severity

10.4.9.2.5.1. Arguments

... objects to be printed

10.4.9.2.6. message(...)

Will add a log entry with message severity

10.4.9.2.6.1. Arguments

... objects to be printed

10.4.9.2.7. info(...)

Will add a log entry with info severity

10.4.9.2.7.1. Arguments

... objects to be printed

10.4.9.2.8. debug(...)

Will add a log entry with debug severity

10.4.9.2.8.1. Arguments

... objects to be printed

10.4.9.2.9. loadfile(filename)

Lua's loadfile() has been modified so that if a file does not exist in the current directory it will look for it in wireshark's user and system directories

10.4.9.2.9.1. Arguments

filename Name of the file to be loaded

10.4.9.2.10. dofile(filename)

Lua's dofile() has been modified so that if a file does not exist in the current directory it will look for it in wireshark's user and system directories

10.4.9.2.10.1. Arguments

filename Name of the file to be run

10.4.9.2.11. persconffile_path([filename])

10.4.9.2.11.1. Arguments

filename (optional) A filename

10.4.9.2.11.2. Returns

The full pathname for a file in the personal configuration directory

10.4.9.2.12. datafile_path([filename])

10.4.9.2.12.1. Arguments

filename (optional)	A filename

10.4.9.2.12.2. Returns

The full pathname for a file in wireshark's configuration directory

10.4.9.2.13. register_stat_cmd_arg(argument, [action])

Register a function to handle a -z option

10.4.9.2.13.1. Arguments

argument	Argument
action (optional)	Action

Appendix A. Files and Folders

A.1. Capture Files

To understand which information will remain available after the captured packets are saved to a capture file, it's helpful to know a bit about the capture file contents.

Wireshark uses the libpcap file format as the default format to save captured packets; this format has existed for a long time and it's pretty simple. However, it has some drawbacks: it's not extensible and lacks some information that would be really helpful (e.g. being able to add a comment to a packet such as "the problems start here" would be really nice).

In addition to the libpcap format, Wireshark supports several different capture file formats. However, the problems described above also applies for these formats.

A new capture file format "PCAP Next Generation Dump File Format" is currently under development, which will fix these drawbacks. However, it still might take a while until the new file format is ready and Wireshark can use it.

A.1.1. Libpcap File Contents

At the start of each libpcap capture file some basic information is stored like a magic number to identify the libpcap file format. The most interesting information of this file start is the link layer type (Ethernet, Token Ring, ...).

The following data is saved for each packet:

- the timestamp with millisecond resolution
- the packet length as it was "on the wire"
- the packet length as it's saved in the file
- the packet's raw bytes

A detailed description of the libpcap file format can be found at: http://wiki.wireshark.org/Development/LibpcapFileFormat

A.1.2. Not Saved in the Capture File

Probably even more interesting for everyday Wireshark usage is to know the things that are **not saved** in the capture file:

- current selections (selected packet, ...)
- name resolution information, see Section 7.7, “Name Resolution” for details

Warning!

The name resolution information is rebuilt each time Wireshark is restarted so this information might even change when the capture file is reopened on the same machine later!

- the number of packets dropped while capturing
- packet marks set with "Edit/Mark Packet"
- time references set with "Edit/Time Reference"
- the current display filter
- ...

A.2. Configuration Files and Folders

Wireshark uses a number of files and folders while it is running. Some of these reside in the personal configuration folder and are used to maintain information between runs of Wireshark, while some of them are maintained in system areas.

Tip

A list of the folders Wireshark actually uses can be found under the **Folders** tab in the dialog box shown when you select **About Wireshark** from the **Help** menu.

The content format of the configuration files is the same on all platforms. However, to match the different policies for Unix and Windows platforms, different folders are used for these files.

Table A.1. Configuration files and folders overview

File/Folder	Description	Unix/Linux folders	Windows folders
preferences	Settings from the Preferences dialog box.	/ etc/wireshark.conf, $HOME/.wireshark/preferences	%WIRESHARK%\wireshark.conf, %APPDATA%\Wireshark\preferences
recent	Recent GUI settings (e.g. recent files lists).	$HOME/.wireshark/recent	%APPDATA%\Wireshark\recent
cfilters	Capture filters.	$HOME/.wireshark/cfilters	%WIRESHARK%\cfilters, %APPDATA%\Wireshark\cfilters
dfilters	Display filters.	$HOME/.wireshark/dfilters	%WIRESHARK%\dfilters, %APPDATA%\Wireshark\dfilters
colorfilters	Coloring rules.	$HOME/.wireshark/colorfilters	%WIRESHARK%\colorfilters, %APPDATA%\Wireshark\colorfilters
disabled_protos	Disabled protocols.	$HOME/.wireshark/disabled_protos	%WIRESHARK%\disabled_protos, %APPDATA%\Wireshark\disabled_protos
ethers	Ethernet name resolution.	/etc/ethers, $HOME/.wireshark/ethers	%WIRESHARK%\ethers, %APPDATA%\Wireshark\ethers
manuf	Ethernet name resolution.	/etc/manuf, $HOME/.wireshark/manuf	%WIRESHARK%\manuf, %APPDATA%\Wireshark\manuf
hosts	IPv4 and IPv6 name resolution.	/etc/hosts, $HOME/.wireshark/hosts	%WIRESHARK%\hosts, %APPDATA%\Wireshark\hosts
services	Network services.	/etc/services, $HOME/.wireshark/services	%WIRESHARK%\services, %APPDATA%\Wireshark\services
subnets	IPv4 subnet name resolution.	/etc/subnets, $HOME/.wireshark/subnets	%WIRESHARK%\subnets, %APPDATA%\Wireshark\subnets
ipxnets	IPX name resolu-	/etc/ipxnets,	%WIRESHARK%\ipxnets,

File/Folder	Description	Unix/Linux folders	Windows folders
	tion.	$HOME/.wireshark/ipxnets	%APPDATA%\Wireshark\ipxnets
plugins	Plugin directories.	/usr/share/wireshark/plugins, /usr/local/share/wireshark/plugins, $HOME/.wireshark/plugins	%WIRESHARK%\plugins\<version>, %APPDATA%\Wireshark\plugins
temp	Temporary files.	Environment: TMPDIR	Environment: TMPDIR or TEMP

Windows folders

%APPDATA% points to the personal configuration folder, e.g.: `C:\Documents and Settings\<username>\Application Data` (details can be found at: Section A.3.1, “Windows profiles”),

%WIRESHARK% points to the Wireshark program folder, e.g.: `C:\Program Files\Wireshark`

Unix/Linux folders

The `/etc` folder is the global Wireshark configuration folder. The folder actually used on your system may vary, maybe something like: `/usr/local/etc`.

$HOME is usually something like: `/home/<username>`

preferences/wireshark.conf

This file contains your Wireshark preferences, including defaults for capturing and displaying packets. It is a simple text file containing statements of the form:

```
variable: value
```

The settings from this file are read in at program start and written to disk when you press the Save button in the "Preferences" dialog box.

recent

This file contains various GUI related settings like the main window position and size, the recent files list and such. It is a simple text file containing statements of the form:

```
variable: value
```

It is read at program start and written at program exit.

cfilters

This file contains all the capture filters that you have defined and saved. It consists of one or more lines, where each line has the following format:

```
"<filter name>" <filter string>
```

The settings from this file are read in at program start and written to disk when you press the Save button in the "Capture Filters" dialog box.

dfilters

This file contains all the display filters that you have defined and saved. It consists of one or more lines, where each line has the following format:

```
"<filter name>" <filter string>
```

The settings from this file are read in at program start and written to disk when you press the Save button in the "Display Filters" dialog box.

colorfilters

This file contains all the color filters that you have defined and saved. It consists of one or more lines, where each line has the following format:

```
@<filter name>@<filter string>@[<bg RGB(16-bit)>][<fg RGB(16-bit)>]
```

The settings from this file are read in at program start and written to disk when you press the Save button in the "Coloring Rules" dialog box.

disabled_protos

Each line in this file specifies a disabled protocol name. The following are some examples:

```
tcp
udp
```

The settings from this file are read in at program start and written to disk when you press the Save button in the "Enabled Protocols" dialog box.

ethers

When Wireshark is trying to translate Ethernet hardware addresses to names, it consults the files listed in Table A.1, “Configuration files and folders overview”. If an address is not found in /etc/ethers, Wireshark looks in $HOME/.wireshark/ethers

Each line in these files consists of one hardware address and name separated by whitespace. The digits of hardware addresses are separated by colons (:), dashes (-) or periods(.). The following are some examples:

```
ff-ff-ff-ff-ff-ff    Broadcast
c0-00-ff-ff-ff-ff    TR_broadcast
00.2b.08.93.4b.a1    Freds_machine
```

The settings from this file are read in at program start and never written by Wireshark.

manuf

Wireshark uses the files listed in Table A.1, “Configuration files and folders overview” to translate the first three bytes of an Ethernet address into a manufacturers name. This file has the same format as the ethers file, except addresses are three bytes long.

An example is:

```
00:00:01 Xerox                  # XEROX CORPORATION
```

The settings from this file are read in at program start and never written by Wireshark.

hosts

Wireshark uses the files listed in Table A.1, “Configuration files and folders overview” to translate IPv4 and IPv6 addresses into names.

This file has the same format as the usual /etc/hosts file on Unix systems.

An example is:

```
# Comments must be prepended by the # sign!
192.168.0.1 homeserver
```

The settings from this file are read in at program start and never written by Wireshark.

services

Wireshark uses the files listed in Table A.1, “Configuration files and folders overview” to translate port numbers into names.

An example is:

```
mydns       5045/udp     # My own Domain Name Server
mydns       5045/tcp     # My own Domain Name Server
```

The settings from this file are read in at program start and never written by Wireshark.

subnets

Wireshark uses the files listed in Table A.1, “Configuration files and folders overview” to translate an IPv4 address into a subnet name. If no exact match from the hosts file or from DNS is found, Wireshark will attempt a partial match for the subnet of the address.

Each line of this file consists of an IPv4 address, a subnet mask length separated only by a '/' and a name separated by whitespace. While the address must be a full IPv4 address, any values beyond the mask length are subsequently ignored.

An example is:

```
# Comments must be prepended by the # sign!
192.168.0.0/24 ws_test_network
```

A partially matched name will be printed as "subnet-name.remaining-address". For example, "192.168.0.1" under the subnet above would be printed as "ws_test_network.1"; if the mask length above had been 16 rather than 24, the printed address would be "ws_test_network.0.1".

The settings from this file are read in at program start and never written by Wireshark.

ipxnets

Wireshark uses the files listed in Table A.1, “Configuration files and folders overview” to translate IPX network numbers into names.

An example is:

```
C0.A8.2C.00      HR
c0-a8-1c-00      CEO
00:00:BE:EF      IT_Server1
110f             FileServer3
```

The settings from this file are read in at program start and never written by Wireshark.

plugins folder

Wireshark searches for plugins in the directories listed in Table A.1, “Configuration files and folders overview”. They are searched in the order listed.

temp folder

If you start a new capture and don't specify a filename for it, Wireshark uses this directory to store that file; see Section 4.7, “Capture files and file modes”.

A.3. Windows folders

Here you will find some details about the folders used in Wireshark on different Windows versions.

As already mentioned, you can find the currently used folders in the **About Wireshark** dialog.

A.3.1. Windows profiles

Windows uses some special directories to store user configuration files which define the "user profile". This can be confusing, as the default directory location changed from Windows version to version and might also be different for English and internationalized versions of Windows.

Note!

If you've upgraded to a new Windows version, your profile might be kept in the former location, so the defaults mentioned here might not apply.

The following guides you to the right place where to look for Wireshark's profile data.

Vista	`C:\Users\<username>\AppData\Roaming\Wireshark`
XP/2000	`C:\Documents and Settings\<username>\Application Data`, "Documents and Settings" and "Application Data" might be internationalized.
NT 4 (no longer supported by Wireshark)	`C:\WINNT\Profiles\<username>\Application Data\Wireshark`
ME/98 - with enabled user profiles (no longer supported by Wireshark)	In Windows ME and 98 you can enable separate user profiles. In that case, something like: `C:\windows\Profiles\<username>\Application Data\Wireshark` is used.
ME/98/95 (no longer supported by Wireshark)	The default in Windows ME/98/95 is: all users work with the same profile, which is located at: `C:\windows\Application Data\Wireshark`

A.3.2. Windows Vista/XP/2000/NT roaming profiles

The following will only be applicable if you are using roaming profiles. This might be the case, if you work in a Windows domain environment (used in company networks). The configurations of all programs you use won't be saved on the local hard drive of the computer you are currently working on, but on the domain server.

As Wireshark is using the correct places to store its profile data, your settings will travel with you, if you logon to a different computer the next time.

There is an exception to this: The "Local Settings" folder in your profile data (typically something like: `C:\Documents and Settings\<username>\Local Settings`) will not be transferred to the domain server. This is the default for temporary capture files.

A.3.3. Windows temporary folder

Wireshark uses the folder which is set by the TMPDIR or TEMP environment variable. This variable will be set by the Windows installer.

Vista	`XXX - could someone give information about this?`
XP/2000	`C:\Documents and Settings\<username>\Local Settings\Temp`
NT 4	`C:\TEMP`

Appendix B. Protocols and Protocol Fields

Wireshark distinguishes between protocols (e.g. tcp) and protocol fields (e.g. tcp.port).

A comprehensive list of all protocols and protocol fields can be found at: http://www.wireshark.org/docs/dfref/

Appendix C. Wireshark Messages

Wireshark provides you with additional information generated out of the plain packet data or it may need to indicate dissection problems. Messages generated by Wireshark are usually placed in [] parentheses.

C.1. Packet List Messages

These messages might appear in the packet list.

C.1.1. [Malformed Packet]

Malformed packet means that the protocol dissector can't dissect the contents of the packet any further. There can be various reasons:

- **Wrong dissector**: Wireshark erroneously has chosen the wrong protocol dissector for this packet. This will happen e.g. if you are using a protocol not on its well known TCP or UDP port. You may try Analyze|Decode As to circumvent this problem.
- **Packet not reassembled**: The packet is longer than a single frame and it is not reassembled, see Section 7.6, “Packet Reassembling” for further details.
- **Packet is malformed**: The packet is actually wrong (malformed), meaning that a part of the packet is just not as expected (not following the protocol specifications).
- **Dissector is buggy**: The corresponding protocol dissector is simply buggy or still incomplete.

Any of the above is possible. You'll have to look into the specific situation to determine the reason. You could disable the dissector by disabling the protocol on the Analyze menu and check how Wireshark displays the packet then. You could (if it's TCP) enable reassembly for TCP and the specific dissector (if possible) in the Edit|Preferences menu. You could check the packet contents yourself by reading the packet bytes and comparing it to the protocol specification. This could reveal a dissector bug. Or you could find out that the packet is indeed wrong.

C.1.2. [Packet size limited during capture]

The packet size was limited during capture, see "Limit each packet to n bytes" at the Section 4.5, “The "Capture Options" dialog box”. While dissecting, the current protocol dissector was simply running out of packet bytes and had to give up. There's nothing else you can do now, except to repeat the whole capture process again with a higher (or no) packet size limitation.

C.2. Packet Details Messages

These messages might appear in the packet details.

C.2.1. [Response in frame: 123]

The current packet is the request of a detected request/response pair. You can directly jump to the corresponding response packet just by double clicking on this message.

C.2.2. [Request in frame: 123]

Same as "Response in frame: 123" above, but the other way round.

C.2.3. [Time from request: 0.123 seconds]

The time between the request and the response packets.

C.2.4. [Stream setup by PROTOCOL (frame 123)]

The session control protocol (SDP, H225, etc) message which signaled the creation of this session. You can directly jump to the corresponding packet just by double clicking on this message.

Appendix D. Related command line tools

D.1. Introduction

Besides the Wireshark GUI application, there are some command line tools which can be helpful for doing some more specialized things. These tools will be described in this chapter.

D.2. tshark: Terminal-based Wireshark

TShark is a terminal oriented version of Wireshark designed for capturing and displaying packets when an interactive user interface isn't necessary or available. It supports the same options as **wireshark**. For more information on **tshark**, see the manual pages (**man tshark**).

D.3. tcpdump: Capturing with tcpdump for viewing with Wireshark

There are occasions when you want to capture packets using **tcpdump** rather than **wireshark**, especially when you want to do a remote capture and do not want the network load associated with running Wireshark remotely (not to mention all the X traffic polluting your capture).

However, the default **tcpdump** parameters result in a capture file where each packet is truncated, because **tcpdump**, by default, only captures the first 68 bytes of each packet.

To ensure that you capture complete packets, use the following command:

```
tcpdump -i <interface> -s 1500 -w <some-file>
```

You will have to specify the correct **interface** and the name of a **file** to save into. In addition, you will have to terminate the capture with ^C when you believe you have captured enough packets.

Note!

tcpdump is not part of the Wireshark distribution. You can get it from: http://www.tcpdump.org for various platforms.

D.4. dumpcap: Capturing with dumpcap for viewing with Wireshark

Dumpcap is a network traffic dump tool. It captures packet data from a live network and writes the packets to a file. Dumpcap's native capture file format is libpcap format, which is also the format used by Wireshark, tcpdump and various other tools.

Without any options set it will use the pcap library to capture traffic from the first available network interface and write the received raw packet data, along with the packets' time stamps into a libpcap file.

Packet capturing is performed with the pcap library. The capture filter syntax follows the rules of the pcap library.

Example D.1. Help information available from dumpcap

```
dumpcap -h
Dumpcap 1.1.4
Capture network packets and dump them into a libpcap file.
See http://www.wireshark.org for more information.

Usage: dumpcap [options] ...

Capture interface:
  -i <interface>           name or idx of interface (def: first non-loopback)
  -f <capture filter>      packet filter in libpcap filter syntax
  -s <snaplen>             packet snapshot length (def: 65535)
  -p                       don't capture in promiscuous mode
  -B <buffer size>         size of kernel buffer (def: 1MB)
  -y <link type>           link layer type (def: first appropriate)
  -D                       print list of interfaces and exit
  -L                       print list of link-layer types of iface and exit
  -S                       print statistics for each interface once every second
  -M                       for -D, -L, and -S produce machine-readable output

Stop conditions:
  -c <packet count>        stop after n packets (def: infinite)
  -a <autostop cond.> ...  duration:NUM - stop after NUM seconds
                           filesize:NUM - stop this file after NUM KB
                              files:NUM - stop after NUM files
Output (files):
  -w <filename>            name of file to save (def: tempfile)
  -b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
                           filesize:NUM - switch to next file after NUM KB
                              files:NUM - ringbuffer: replace after NUM files
  -n                       use pcapng format instead of pcap
Miscellaneous:
  -v                       print version information and exit
  -h                       display this help and exit

Example: dumpcap -i eth0 -a duration:60 -w output.pcap
"Capture network packets from interface eth0 until 60s passed into output.pcap"

Use Ctrl-C to stop capturing at any time.
```

D.5. capinfos: Print information about capture files

Included with Wireshark is a small utility called **capinfos**, which is a command-line utility to print information about binary capture files.

Example D.2. Help information available from capinfos

```
$ capinfos -h
Capinfos 1.1.4
Prints information about capture files.
See http://www.wireshark.org for more information.

Usage: capinfos [options] <infile> ...

General:
  -t display the capture file type
  -E display the capture file encapsulation

Size:
  -c display the number of packets
  -s display the size of the file (in bytes)
  -d display the total length of all packets (in bytes)

Time:
  -u display the capture duration (in seconds)
  -a display the capture start time
  -e display the capture end time

Statistic:
  -y display average data rate (in bytes/sec)
  -i display average data rate (in bits/sec)
  -z display average packet size (in bytes)
  -x display average packet rate (in packets/sec)

Miscellaneous:
  -h display this help and exit

If no options are given the default is to display all infos
```

D.6. editcap: Edit capture files

Included with Wireshark is a small utility called **editcap**, which is a command-line utility for working with capture files. Its main function is to remove packets from capture files, but it can also be used to convert capture files from one format to another, as well as to print information about capture files.

Example D.3. Help information available from editcap

```
$ editcap -h
Editcap 1.1.4
Edit and/or translate the format of capture files.
See http://www.wireshark.org for more information.

Usage: editcap [options] ... <infile> <outfile> [ <packet#>[-<packet#>] ... ]

<infile> and <outfile> must both be present.
A single packet or a range of packets can be selected.

Packet selection:
  -r                       keep the selected packets; default is to delete them.
  -A <start time>          don't output packets whose timestamp is before the
                           given time (format as YYYY-MM-DD hh:mm:ss).
  -B <stop time>           don't output packets whose timestamp is after the
                           given time (format as YYYY-MM-DD hh:mm:ss).

Duplicate packet removal:
  -d                       remove packet if duplicate (window == 5).
  -D <dup window>          remove packet if duplicate; configurable <dup window>
                           Valid <dup window> values are 0 to 1000000.
                           NOTE: A <dup window> of 0 with -v (verbose option) is
                           useful to print MD5 hashes.
  -w <dup time window>     remove packet if duplicate packet is found EQUAL TO OR
                           LESS THAN <dup time window> prior to current packet.
                           A <dup time window> is specified in relative seconds
                           (e.g. 0.000001).

            NOTE: The use of the 'Duplicate packet removal' options with
            other editcap options except -v may not always work as expected.
            Specifically the -r and -t options will very likely NOT have the
            desired effect if combined with the -d, -D or -w.

Packet manipulation:
  -s <snaplen>             truncate each packet to max. <snaplen> bytes of data.
  -C <choplen>             chop each packet at the end by <choplen> bytes.
  -t <time adjustment>     adjust the timestamp of each packet;
                           <time adjustment> is in relative seconds (e.g. -0.5).
  -E <error probability>   set the probability (between 0.0 and 1.0 incl.)
                           that a particular packet byte will be randomly changed.

Output File(s):
  -c <packets per file>    split the packet output to different files
                           based on uniform packet counts
                           with a maximum of <packets per file> each.
  -i <seconds per file>    split the packet output to different files
                           based on uniform time intervals
                           with a maximum of <seconds per file> each.
  -F <capture type>        set the output file type; default is libpcap.
                           an empty "-F" option will list the file types.
  -T <encap type>          set the output file encapsulation type;
                           default is the same as the input file.
                           an empty "-T" option will list the encapsulation types.

Miscellaneous:
  -h                       display this help and exit.
  -v                       verbose output.
                           If -v is used with any of the 'Duplicate Packet
                           Removal' options (-d, -D or -w) then Packet lengths
                           and MD5 hashes are printed to standard-out.

$ editcap -F
editcap: option requires an argument -- F
```

```
editcap: The available capture file types for "F":
    libpcap - Wireshark/tcpdump/... - libpcap
    nseclibpcap - Wireshark - nanosecond libpcap
    modlibpcap - Modified tcpdump - libpcap
    nokialibpcap - Nokia tcpdump - libpcap
    rh6_1libpcap - RedHat 6.1 tcpdump - libpcap
    suse6_3libpcap - SuSE 6.3 tcpdump - libpcap
    5views - Accellent 5Views capture
    dct2000 - Catapult DCT2000 trace (.out format)
    nettl - HP-UX nettl trace
    netmon1 - Microsoft NetMon 1.x
    netmon2 - Microsoft NetMon 2.x
    ngsniffer - NA Sniffer (DOS)
    ngwsniffer_1_1 - NA Sniffer (Windows) 1.1
    ngwsniffer_2_0 - NA Sniffer (Windows) 2.00x
    niobserverv9 - Network Instruments Observer (V9)
    lanalyzer - Novell LANalyzer
    snoop - Sun snoop
    rf5 - Tektronix K12xx 32-bit .rf5 format
    visual - Visual Networks traffic capture
    k12text - K12 text file
    commview - TamoSoft CommView
    pcapng - Wireshark - pcapng (experimental)

$ editcap -T
editcap: option requires an argument -- T
editcap: The available encapsulation types for "T":
    ether - Ethernet
    tr - Token Ring
    slip - SLIP
    ppp - PPP
    fddi - FDDI
    fddi-swapped - FDDI with bit-swapped MAC addresses
    rawip - Raw IP
    arcnet - ARCNET
    arcnet_linux - Linux ARCNET
    atm-rfc1483 - RFC 1483 ATM
    linux-atm-clip - Linux ATM CLIP
    lapb - LAPB
    atm-pdus - ATM PDUs
    atm-pdus-untruncated - ATM PDUs - untruncated
    null - NULL
    ascend - Lucent/Ascend access equipment
    isdn - ISDN
    ip-over-fc - RFC 2625 IP-over-Fibre Channel
    ppp-with-direction - PPP with Directional Info
    ieee-802-11 - IEEE 802.11 Wireless LAN
    prism - IEEE 802.11 plus Prism II monitor mode header
    ieee-802-11-radio - IEEE 802.11 Wireless LAN with radio information
    ieee-802-11-radiotap - IEEE 802.11 plus radiotap WLAN header
    ieee-802-11-avs - IEEE 802.11 plus AVS WLAN header
    linux-sll - Linux cooked-mode capture
    frelay - Frame Relay
    frelay-with-direction - Frame Relay with Directional Info
    chdlc - Cisco HDLC
    ios - Cisco IOS internal
    ltalk - Localtalk
    pflog-old - OpenBSD PF Firewall logs, pre-3.4
    hhdlc - HiPath HDLC
    docsis - Data Over Cable Service Interface Specification
    cosine - CoSine L2 debug log
    whdlc - Wellfleet HDLC
    sdlc - SDLC
    tzsp - Tazmen sniffer protocol
    enc - OpenBSD enc(4) encapsulating interface
    pflog - OpenBSD PF Firewall logs
    chdlc-with-direction - Cisco HDLC with Directional Info
    bluetooth-h4 - Bluetooth H4
    mtp2 - SS7 MTP2
    mtp3 - SS7 MTP3
    irda - IrDA
    user0 - USER 0
    user1 - USER 1
    user2 - USER 2
    user3 - USER 3
    user4 - USER 4
    user5 - USER 5
    user6 - USER 6
    user7 - USER 7
    user8 - USER 8
```

```
user9 - USER 9
user10 - USER 10
user11 - USER 11
user12 - USER 12
user13 - USER 13
user14 - USER 14
user15 - USER 15
symantec - Symantec Enterprise Firewall
ap1394 - Apple IP-over-IEEE 1394
bacnet-ms-tp - BACnet MS/TP
raw-icmp-nettl - Raw ICMP with nettl headers
raw-icmpv6-nettl - Raw ICMPv6 with nettl headers
gprs-llc - GPRS LLC
juniper-atm1 - Juniper ATM1
juniper-atm2 - Juniper ATM2
redback - Redback SmartEdge
rawip-nettl - Raw IP with nettl headers
ether-nettl - Ethernet with nettl headers
tr-nettl - Token Ring with nettl headers
fddi-nettl - FDDI with nettl headers
unknown-nettl - Unknown link-layer type with nettl headers
mtp2-with-phdr - MTP2 with pseudoheader
juniper-pppoe - Juniper PPPoE
gcom-tie1 - GCOM TIE1
gcom-serial - GCOM Serial
x25-nettl - X25 with nettl headers
k12 - K12 protocol analyzer
juniper-mlppp - Juniper MLPPP
juniper-mlfr - Juniper MLFR
juniper-ether - Juniper Ethernet
juniper-ppp - Juniper PPP
juniper-frelay - Juniper Frame-Relay
juniper-chdlc - Juniper C-HDLC
juniper-ggsn - Juniper GGSN
lapd - LAPD
dct2000 - Catapult DCT2000
ber - ASN.1 Basic Encoding Rules
juniper-vp - Juniper Voice PIC
usb - Raw USB packets
ieee-802-16-mac-cps - IEEE 802.16 MAC Common Part Sublayer
raw-telnet-nettl - Raw telnet with nettl headers
usb-linux - USB packets with Linux header
mpeg - MPEG
ppi - Per-Packet Information header
erf - Endace Record File
bluetooth-h4 - Bluetooth H4 with linux header
sita-wan - SITA WAN packets
sccp - SS7 SCCP
bluetooth-hci - Bluetooth without transport layer
ipmb - Intelligent Platform Management Bus
wpan - IEEE 802.15.4 Wireless PAN
x2e-xoraya - X2E Xoraya
flexray - FlexRay
lin - Local Interconnect Network
most - Media Oriented Systems Transport
can20b - Controller Area Network 2.0B
layer1-event - EyeSDN Layer 1 event
x2e-serial - X2E serial line capture
i2c - I2C
wpan-nonask-phy - IEEE 802.15.4 Wireless PAN non-ASK PHY
tnef - Transport-Neutral Encapsulation Format
usb-linux-mmap - USB packets with Linux header and padding
gsm_um - GSM Um Interface
```

D.7. mergecap: Merging multiple capture files into one

Mergecap is a program that combines multiple saved capture files into a single output file specified by the -w argument. Mergecap knows how to read libpcap capture files, including those of tcpdump. In addition, Mergecap can read capture files from snoop (including Shomiti) and atmsnoop, LanAlyzer, Sniffer (compressed or uncompressed), Microsoft Network Monitor, AIX's iptrace, NetXray, Sniffer Pro, RADCOM's WAN/LAN analyzer, Lucent/Ascend router debug output, HP-UX's nettl, and the dump output from Toshiba's ISDN routers. There is no need to tell Mergecap what type of file you are reading; it will determine the file type by itself. Mergecap is also capable of reading any of these file formats if they are compressed using gzip. Mergecap recognizes this directly from the file; the '.gz' extension is not required for this purpose.

By default, it writes the capture file in libpcap format, and writes all of the packets in the input capture files to the output file. The -F flag can be used to specify the format in which to write the capture file; it can write the file in libpcap format (standard libpcap format, a modified format used by some patched versions of libpcap, the format used by Red Hat Linux 6.1, or the format used by SuSE Linux 6.3), snoop format, uncompressed Sniffer format, Microsoft Network Monitor 1.x format, and the format used by Windows-based versions of the Sniffer software.

Packets from the input files are merged in chronological order based on each frame's timestamp, unless the -a flag is specified. Mergecap assumes that frames within a single capture file are already stored in chronological order. When the -a flag is specified, packets are copied directly from each input file to the output file, independent of each frame's timestamp.

If the -s flag is used to specify a snapshot length, frames in the input file with more captured data than the specified snapshot length will have only the amount of data specified by the snapshot length written to the output file. This may be useful if the program that is to read the output file cannot handle packets larger than a certain size (for example, the versions of snoop in Solaris 2.5.1 and Solaris 2.6 appear to reject Ethernet frames larger than the standard Ethernet MTU, making them incapable of handling gigabit Ethernet captures if jumbo frames were used).

If the -T flag is used to specify an encapsulation type, the encapsulation type of the output capture file will be forced to the specified type, rather than being the type appropriate to the encapsulation type of the input capture file. Note that this merely forces the encapsulation type of the output file to be the specified type; the packet headers of the packets will not be translated from the encapsulation type of the input capture file to the specified encapsulation type (for example, it will not translate an Ethernet capture to an FDDI capture if an Ethernet capture is read and '-T fddi' is specified).

Example D.4. Help information available from mergecap

```
$ mergecap -h
Mergecap 1.1.4
Merge two or more capture files into one.
See http://www.wireshark.org for more information.

Usage: mergecap [options] -w <outfile>|- <infile> ...

Output:
  -a                concatenate rather than merge files.
                    default is to merge based on frame timestamps.
  -s <snaplen>      truncate packets to <snaplen> bytes of data.
  -w <outfile>|-    set the output filename to <outfile> or '-' for stdout.
  -F <capture type> set the output file type; default is libpcap.
                    an empty "-F" option will list the file types.
  -T <encap type>   set the output file encapsulation type;
                    default is the same as the first input file.
                    an empty "-T" option will list the encapsulation types.
```

```
Miscellaneous:
  -h                display this help and exit.
  -v                verbose output.
```

A simple example merging `dhcp-capture.libpcap` and `imap-1.libpcap` into `out-file.libpcap` is shown below.

Example D.5. Simple example of using mergecap

```
$ mergecap -w outfile.libpcap dhcp-capture.libpcap imap-1.libpcap
```

D.8. text2pcap: Converting ASCII hexdumps to network captures

There may be some occasions when you wish to convert a hex dump of some network traffic into a libpcap file.

Text2pcap is a program that reads in an ASCII hex dump and writes the data described into a libpcap-style capture file. text2pcap can read hexdumps with multiple packets in them, and build a capture file of multiple packets. text2pcap is also capable of generating dummy Ethernet, IP and UDP headers, in order to build fully processable packet dumps from hexdumps of application-level data only.

Text2pcap understands a hexdump of the form generated by od -A x -t x1. In other words, each byte is individually displayed and surrounded with a space. Each line begins with an offset describing the position in the file. The offset is a hex number (can also be octal - see -o), of more than two hex digits. Here is a sample dump that text2pcap can recognize:

```
000000 00 e0 1e a7 05 6f 00 10 ........
000008 5a a0 b9 12 08 00 46 00 ........
000010 03 68 00 00 00 00 0a 2e ........
000018 ee 33 0f 19 08 7f 0f 19 ........
000020 03 80 94 04 00 00 10 01 ........
000028 16 a2 0a 00 03 50 00 0c ........
000030 01 01 0f 19 03 80 11 01 ........
```

There is no limit on the width or number of bytes per line. Also the text dump at the end of the line is ignored. Bytes/hex numbers can be uppercase or lowercase. Any text before the offset is ignored, including email forwarding characters '>'. Any lines of text between the bytestring lines is ignored. The offsets are used to track the bytes, so offsets must be correct. Any line which has only bytes without a leading offset is ignored. An offset is recognized as being a hex number longer than two characters. Any text after the bytes is ignored (e.g. the character dump). Any hex numbers in this text are also ignored. An offset of zero is indicative of starting a new packet, so a single text file with a series of hexdumps can be converted into a packet capture with multiple packets. Multiple packets are read in with timestamps differing by one second each. In general, short of these restrictions, text2pcap is pretty liberal about reading in hexdumps and has been tested with a variety of mangled outputs (including being forwarded through email multiple times, with limited line wrap etc.)

There are a couple of other special features to note. Any line where the first non-whitespace character is '#' will be ignored as a comment. Any line beginning with #TEXT2PCAP is a directive and options can be inserted after this command to be processed by text2pcap. Currently there are no directives implemented; in the future, these may be used to give more fine grained control on the dump and the way it should be processed e.g. timestamps, encapsulation type etc.

Text2pcap also allows the user to read in dumps of application-level data, by inserting dummy L2, L3 and L4 headers before each packet. Possiblities include inserting headers such as Ethernet, Ethernet + IP, Ethernet + IP + UDP, or Ethernet + Ip + TCP before each packet. This allows Wireshark or any other full-packet decoder to handle these dumps.

Example D.6. Help information available for text2pcap

```
$ text2pcap -h
Text2pcap 1.1.4
Generate a capture file from an ASCII hexdump of packets.
See http://www.wireshark.org for more information.

Usage: text2pcap [options] <infile> <outfile>
```

```
where  <infile> specifies input  filename (use - for standard input)
      <outfile> specifies output filename (use - for standard output)

Input:
  -o hex|oct|dec         parse offsets as (h)ex, (o)ctal or (d)ecimal; default is hex.
  -t <timefmt>           treats the text before the packet as a date/time code;
                         the specified argument is a format string of the sort
                         supported by strptime.
                         Example: The time "10:15:14.5476" has the format code
                         "%H:%M:%S."
                         NOTE: The subsecond component delimiter must be given
                          (.) but no pattern is required; the remaining number
                          is assumed to be fractions of a second.
                         NOTE: Date/time fields from the current date/time are
                         used as the default for unspecified fields.

Output:
  -l <typenum>           link-layer type number; default is 1 (Ethernet).
                         See the file net/bpf.h for list of numbers.
                         Use this option if your dump is a complete hex dump
                         of an encapsulated packet and you wish to specify
                         the exact type of encapsulation.
                         Example: -l 7 for ARCNet packets.
  -m <max-packet>        max packet length in output; default is 64000

Prepend dummy header:
  -e <l3pid>             prepend dummy Ethernet II header with specified L3PID
                         (in HEX).
                         Example: -e 0x806 to specify an ARP packet.
  -i <proto>             prepend dummy IP header with specified IP protocol
                         (in DECIMAL).
                         Automatically prepends Ethernet header as well.
                         Example: -i 46
  -u <srcp>,<destp>      prepend dummy UDP header with specified
                         dest and source ports (in DECIMAL).
                         Automatically prepends Ethernet & IP headers as well.
                         Example: -u 1000 69 to make the packets look like TFTP/UDP packets.
  -T <srcp>,<destp>      prepend dummy TCP header with specified
                         dest and source ports (in DECIMAL).
                         Automatically prepends Ethernet & IP headers as well.
                         Example: -T 50,60
  -s <srcp>,<dstp>,<tag> prepend dummy SCTP header with specified
                         dest/source ports and verification tag (in DECIMAL).
                         Automatically prepends Ethernet & IP headers as well.
                         Example: -s 30,40,34
  -S <srcp>,<dstp>,<ppi> prepend dummy SCTP header with specified
                         dest/source ports and verification tag 0.
                         Automatically prepends a dummy SCTP DATA
                         chunk header with payload protocol identifier ppi.
                         Example: -S 30,40,34

Miscellaneous:
  -h                     display this help and exit.
  -d                     detailed debug of parser states.
  -q                     generate no output at all (automatically turns off -d).
```

D.9. idl2wrs: Creating dissectors from CORBA IDL files

In an ideal world idl2wrs would be mentioned in the users guide in passing and documented in the developers guide. As the developers guide has not yet been completed it will be documented here.

D.9.1. What is it?

As you have probably guessed from the name, **idl2wrs** takes a user specified IDL file and attempts to build a dissector that can decode the IDL traffic over GIOP. The resulting file is "C" code, that should compile okay as a Wireshark dissector.

idl2wrs basically parses the data struct given to it by the omniidl compiler, and using the GIOP API available in packet-giop.[ch], generates get_CDR_xxx calls to decode the CORBA traffic on the wire.

It consists of 4 main files.

`README.idl2wrs`	This document
`wireshark_be.py`	The main compiler backend
`wireshark_gen.py`	A helper class, that generates the C code.
`idl2wrs`	A simple shell script wrapper that the end user should use to generate the dissector from the IDL file(s).

D.9.2. Why do this?

It is important to understand what CORBA traffic looks like over GIOP/IIOP, and to help build a tool that can assist in troubleshooting CORBA interworking. This was especially the case after seeing a lot of discussions about how particular IDL types are represented inside an octet stream.

I have also had comments/feedback that this tool would be good for say a CORBA class when teaching students what CORBA traffic looks like "on the wire".

It is also COOL to work on a great Open Source project such as the case with "Wireshark" (http://www.wireshark.org)

D.9.3. How to use idl2wrs

To use the idl2wrs to generate Wireshark dissectors, you need the following:

Prerequisites to using idl2wrs

1. Python must be installed. See http://python.org/
2. omniidl from the the omniORB package must be available. See http://omniorb.sourceforge.net/
3. Of course you need Wireshark installed to compile the code and tweak it if required. idl2wrs is part of the standard Wireshark distribution

To use idl2wrs to generate an Wireshark dissector from an idl file use the following procedure:

Procedure for converting a CORBA idl file into a Wireshark dissector

1. To write the C code to stdout.

   ```
   idl2wrs  <your file.idl>
   ```

   e.g.:

   ```
   idl2wrs echo.idl
   ```

2. To write to a file, just redirect the output.

   ```
   idl2wrs echo.idl > packet-test-idl.c
   ```

   You may wish to comment out the register_giop_user_module() code and that will leave you with heuristic dissection.

If you don't want to use the shell script wrapper, then try steps 3 or 4 instead.

3. To write the C code to stdout.

   ```
   Usage: omniidl  -p ./ -b wireshark_be <your file.idl>
   ```

   e.g.:

   ```
   omniidl  -p ./ -b wireshark_be echo.idl
   ```

4. To write to a file, just redirect the output.

   ```
   omniidl  -p ./ -b wireshark_be echo.idl > packet-test-idl.c
   ```

   You may wish to comment out the register_giop_user_module() code and that will leave you with heuristic dissection.

5. Copy the resulting C code to subdirectory epan/dissectors/ inside your Wireshark source directory.

   ```
   cp packet-test-idl.c /dir/where/wireshark/lives/epan/dissectors/
   ```

   The new dissector has to be added to Makefile.common in the same directory. Look for the declaration CLEAN_DISSECTOR_SRC and add the new dissector there. For example,

   ```
   CLEAN_DISSECTOR_SRC = \
           packet-2dparityfec.c    \
           packet-3com-njack.c     \
           ...
   ```

   becomes

   ```
   CLEAN_DISSECTOR_SRC = \
           packet-test-idl.c       \
           packet-2dparityfec.c    \
           packet-3com-njack.c     \
           ...
   ```

   For the next steps, go up to the top of your Wireshark source directory.

6. Run configure

   ```
   ./configure (or ./autogen.sh)
   ```

7. Compile the code

   ```
   make
   ```

8. Good Luck !!

D.9.4. TODO

1. Exception code not generated (yet), but can be added manually.
2. Enums not converted to symbolic values (yet), but can be added manually.
3. Add command line options etc
4. More I am sure :-)

D.9.5. Limitations

See the TODO list inside `packet-giop.c`

D.9.6. Notes

1. The "-p ./" option passed to omniidl indicates that the wireshark_be.py and wireshark_gen.py are residing in the current directory. This may need tweaking if you place these files somewhere else.
2. If it complains about being unable to find some modules (e.g. tempfile.py), you may want to check if PYTHONPATH is set correctly. On my Linux box, it is PYTHONPATH=/usr/lib/python2.4/

Appendix E. This Document's License (GPL)

As with the original licence and documentation distributed with Wireshark, this document is covered by the GNU General Public Licence (GNU GPL).

If you haven't read the GPL before, please do so. It explains all the things that you are allowed to do with this code and documentation.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
```

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that

system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
```

```
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA


Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year  name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```